uc3m | Universidad Carlos III de Madrid

Grado Universitario en Ingeniería Informática

(2018-2019)

Trabajo Fin de Grado

"Uso de Deep Learning para la digitalización de documentos históricos"

Luis Enrique Güendián García

Tutor: Jose María Valls Ferrán

Co-Tutor: Ricardo Aler Mur

Salón de Grados del Campus de Colmenarejo en la fecha: 04/07/2019

AGRADECIMIENTOS

Para empezar, me gustaría agradecer a aquellas personas que me han ayudado durante la carrera y en especial en este proyecto.

Mención especial a mi familia, que han hecho posible que yo esté aquí, ofreciéndome todo lo posible para que pueda estudiar lo que yo deseo para mi vida profesional.

Por otro lado, es estoy muy agradecido a mis tutores y a Manuel Ayuso García de los cuales he recibido un trato inmejorable y una ayuda perfecta para poder realizar este trabajo.

Y, por último, como no, a aquellos amigos que al principio de esta carrera empezaron siendo compañeros. Los cuales me han facilitado la carrera muchísimo y han sido un apoyo indispensable para poder superarla.

ÍNDICE

ÍNDICE DE IMÁGENES

ÍNDICE DE TABLAS

ÍNDICE DE GRÁFICAS

ÍNDICE DE ECUACIONES

RESUMEN

En los últimos años la digitalización de diferentes documentos [1] ha avanzado mucho, pero no con todos los documentos se obtienen resultados viables. Con los documentos antiguos los resultados no son lo suficientemente buenos a pesar de contar con distintos softwares especializados en este terreno.

En este trabajo se va a abordar la problemática de la digitalización de documentos antiguos, estos tienen diferentes problemáticas sobre los documentos modernos que complican la obtención de buenos resultados.

1. PALABRAS CLAVE

Deep Learning: en español aprendizaje profundo, es una técnica de aprendizaje automático que enseña a las maquinas a aprender mediante ejemplos.

Reconocimiento Óptico de Caracteres (OCR): es una tecnología que permite convertir documentos en diferentes formatos, ya sea archivos PDF o imágenes en datos editables y con opción de búsqueda.

Modelo: es una red de neuronas ya entrenada con características propias y que se utiliza para usar esa red con otro conjunto de datos para obtener la eficacia del ajuste.

Ground truth: es un archivo de texto que incluye la transcripción de una línea del conjunto de datos. Se emplea tanto para realizar el entrenamiento de la red como para la evaluación de esta.

Carácter: es una letra que se puede encontrar en el conjunto de datos utilizado.

Glifo: Forma impresa de uno o varios caracteres y sus rasgos diacríticos que forman una unidad.

2. INTRODUCCIÓN

En esta sección se trata presentar de una manera clara las necesidades y motivos que han llegado a plantear el siguiente Trabajo de Fin de Grado, incluyendo también los objetivos del mismo.

2.1. Motivaciones de este trabajo

En este trabajo se va a abordar la problemática de la digitalización de documentos antiguos, que tienen características específicas bastante diferentes de la digitalización de documentos modernos. Estos documentos tienen una problemática específica como el ruido, debido al deterioro sufrido; la existencia de glosas que en las sucesivas copias se iban incorporando a los documentos; las letras capitales, etc.

Existen herramientas de digitalización orientadas a este tipo de documentos, algunas de ellas basadas en Redes de Neuronas que utilizan técnicas de *Deep Learning* para su aprendizaje. Una de las herramientas más utilizadas es Ocropus que utiliza a su vez redes de tipo LSTM. Dichas redes son capaces de aprender a transcribir documentos si se les proporciona un conjunto de ejemplos de entrenamiento, compuesto de las imágenes del documento y una transcripción a texto de las mismas. En este trabajo se estudiarán distintos aspectos de la digitalización de documentos antiguos con la herramienta Ocropus.

Una primera cuestión está relacionada con que las redes LSTM tienen varios parámetros ajustables que influyen en su aprendizaje, como el número de unidades ocultas, el número de ciclos de entrenamiento y la razón de aprendizaje. La selección de los valores más adecuados de estos parámetros es un asunto de gran importancia que será abordado en este trabajo.

Generalmente se parte de imágenes de las páginas de los documentos. De cada página se deben extraer las diferentes líneas que constituirán los ejemplos con los que se entrenará la red. A cada ejemplo de entrada (imagen de una línea) le corresponderá la transcripción de esa línea en forma de texto. Esta es una tarea muy costosa que debe realizar un experto de forma manual. Lo ideal sería que el experto transcribiera muy pocas líneas pero que fueran suficientes para entrenar de forma adecuada la red. Después, de forma automática, la red podría digitalizar el documento completo. Así, se pretende estudiar la relación del tamaño del conjunto de entrenamiento con los resultados obtenidos, con el objetivo de minimizar el esfuerzo del transcriptor humano.

Entrenar estas redes desde cero requiere un gran coste computacional. Si en lugar de esto se parte de redes que se han entrenado con otros documentos similares, presumiblemente el coste será menor. En este trabajo se explorará la técnica conocida como Transfer Learning, que consiste en utilizar redes pre-entrenadas en una tarea (un documento) como punto de partida para resolver otra (otro documento ligeramente distinto, en este caso).

Todos estos problemas han motivado la realización de este trabajo, que tiene los objetivos que se especifican a continuación.

2.2. Objetivos

Como se ha comentado, a la hora de digitalizar documentos históricos no se cuenta con los mejores resultados posibles, es por ello por lo que el principal objetivo que se persigue con este proyecto es el estudio del comportamiento de redes de neuronas a partir de diferentes parámetros para la digitalización de estos documentos. Además del principal objetivo se cuenta con varios secundarios:

- Utilización de una herramienta especializada en digitalización de documentos antiguos.
- Optimización de esta herramienta ajustando sus parámetros
- Determinación del tamaño del conjunto de entrenamiento mínimo capaz de producir resultados aceptables
- Estudio de técnicas de Transfer Learning para intentar mejorar el aprendizaje de la red, tanto en coste computacional como en resultados.

Con estos objetivos se persigue acercar un poco más el reconocimiento óptico de caracteres u OCR, a este tipo de documentos a la máxima efectividad posible, que claramente no es la que se obtiene actualmente.

2.3. Estructura del documento

En este punto se va a exponer de manera resumida de que trata cada capítulo del documento sobre el trabajo realizado.

En este documento, primero (capítulo 3) se hablará sobre el estado actual de los temas que abarca el proyecto, desde las redes de neuronas a los documentos históricos, pasando por la digitalización de estos.

Después (capítulo 4) se hablará sobre los primeros pasos con el software utilizado, también en este punto se documentarán los resultados de los primeros experimentos realizados para observar el comportamiento de los modelos con diferentes valores en los parámetros.

Seguido (capítulo 5) encontraremos el cuerpo del proyecto, en el cual encontramos el principal estudio del proyecto y de la elección del modelo óptimo, así como de tratar minimizar el conjunto de entrenamiento.

A continuación (capítulo 6) se realizará un estudio con otro documento utilizando Transfer Learning para el proceso de aprendizaje del mismo. En este punto veremos desde la preparación de los datos hasta el análisis de los resultados obtenidos.

En el capítulo 7 encontraremos tanto la planificación como el entorno socioeconómico del proyecto. A su vez en los dos últimos capítulos (capítulo 8 y 9) contaremos con una pequeña conclusión del proyecto y análisis de los objetivos y la bibliografía empleada en este documento.

En los anexos encontraremos los distintos resultados obtenidos de una manera más extendida siguiendo el formato de tabla para una mejor comprensión de los mismos.

2.4. Marco Regulador

En este apartado se va a tratar de hablar sobre los asuntos legales que pueden afectar en este proyecto durante su desarrollo.

2.4.1. Legislación aplicable

Todo el software empleado para realizar este proyecto es libre menos Microsoft Windows 10 que se debe disponer de licencia, por lo que además de esto no aplica ningún tipo de regulación en este apartado. Además, hay que tener en cuenta que al utilizar este software se acepta un acuerdo de licencias con los mismos. Los distintos softwares empleados en el proyecto son:

- Microsoft Windows 10
- Python
- Ocropus

En cuanto a los documentos empleados para las diferentes pruebas realizadas en este proyecto estos deben estar regulados por la La Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales. Es una ley española que tiene como objetivo adaptar el derecho español al Reglamento General de Proteccion de Datos aprobado por la Unión Europea que además de tener que respetar el tratamiento de los datos personales, algo que no concierne a este proyecto, debe garantizar los derechos digitales de estos. Es decir, los documentos utilizados deben cumplir y respetar estos derechos digitales para su correcto uso.

2.4.2. Estándares técnicos

El único lenguaje de programación empleado durante este proyecto es Python, y no tiene un papel protagonista en el mismo, debido a esto no se ha decidido seguir ningún estándar, aunque si se ha seguido un orden básico en el código para su mejor comprensión

2.4.3. Propiedad intelectual

No se va a proteger la propiedad intelectual de este proyecto, ya que las pruebas realizadas no son propias y el proyecto no es patentable. Además, el objetivo que se persigue es el de analizar el problema para ver posibles soluciones y no algo finalista.

Cualquier persona que lo desee podrá usar este estudio para compartirlo o partir de él para diferentes proyectos relacionados con el tema.

3. ESTADO DEL ARTE

En este capítulo de la memoria se va a hablar sobre el estado actual del tema principal del proyecto, los documentos históricos y las redes neuronales y también de la digitalización, tanto de documentos actuales como de documentos históricos.

3.1. Documentos del proyecto

En este punto vamos a hablar sobre el tipo de documentos que se utilizaran en este proyecto. Estos documentos fueron aportados por Manuel Ayuso García, doctor en Filología Clásica, que también realizó manualmente todas las transcripciones necesarias para este trabajo. Dichos documentos son llamados documentos antiguos y representan a aquellos documentos impresos antes de la invención de la prensa automática en el siglo XIX. [2]

3.1.1. Imprenta

La imprenta supuso una gran revolución, fue inventada por Gutenberg, en torno al 1450 imprimió su primer libro, la Biblia de Gutenberg. A España llegó en el 1472.

En esta época las imprentas se caracterizaban por:

- La prensa de tipos móviles que combinaba letrerías de moldes de metal con tacos xilográficos de madera para las ilustraciones.
  Estos materiales pasaban de un impreso a otro causando deterioro por el uso y provocando impresos irregulares.
- La prensa manual -que ha estado vigente hasta la mitad del s. XIX- aplicaba una presión no uniforme sobre el papel que podía ser fabricado de diversos materiales. En la primera época de la imprenta era una mezcla de cuerdas, trapos y telas.
- Los espacios entre letras y palabras eran totalmente variables, ya que constaba de un trabajo manual que causaba pequeños fallos

Tanto la prensa manual como de tipos móviles no son incompatibles, durante los primeros siglos una imprenta podía ser ambas.

Los libros impresos durante el siglo XV se les conocen como incunables de los que actualmente se conservan unos 300000 ejemplares. Estos libros contaban con diferentes características, las principales son:

- Imitación: En los primeros impresos se buscaba que guardaran la mayor semejanza posible con los manuscritos.
- Márgenes: La amplitud de las márgenes permitía la escritura de apostillas o notas al margen.
- Separar párrafos: Se utilizan signos llamados calderones para separar los párrafos.
- Letras capitales: No se incluían letras iniciales porque los impresores de esa época dejaban un hueco al iniciar una obra o capítulo, que posteriormente se dibujaban iniciales llamadas letras capitales.

Es en el año 1510 cuando se realiza la impresión de *Las Sátiras* de Persio en Logroño, documento protagonista durante este proyecto y del que se va a hablar un poco más en los siguientes puntos.

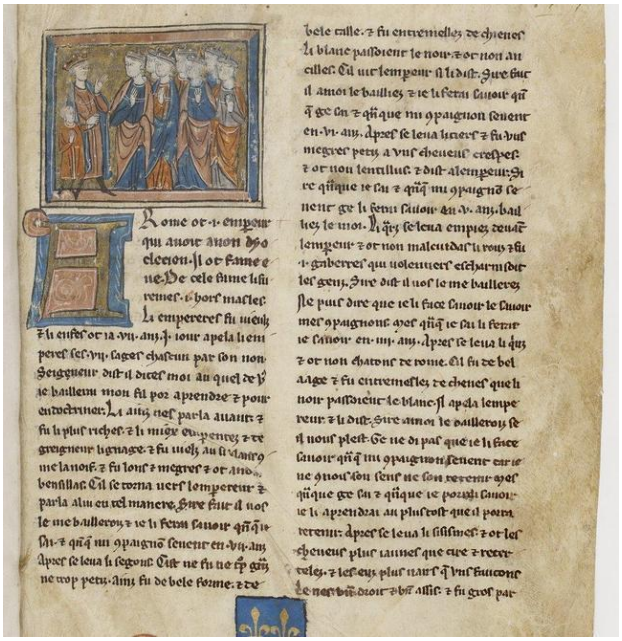


Ilustración 1: Ejemplo de una página de un incunable [3]

3.1.2. Arnao Guillén de Brocar

Este impreso fue realizado en el taller de Arnao Guillén de Brocar en Logroño [4], el de Arnao fue uno de los talleres más importantes y activo en España entre 1490 y 1524. Pasó por Pamplona, Logroño, Valladolid y sobre todo Alcalá de Henares donde sirvió como prensa de la Universidad.

3.1.3. Las Sátiras de Persio

Esta obra fue compuesta por el poeta latino de la *Edad de plata* (14-200 d. C.), Persio. *Las Sátiras* es la única obra que se conversa y está compuesta por seis poemas y un breve prólogo que hacen unos 700 versos, en total constan de 22 páginas en el modelo de trabajo utilizado en este proyecto.

Su obra fue admirada por sus contemporáneos, pero también calificada desde el inicio como obscura y de difícil comprensión.

Estos factores contribuyeron a que la obra fuera bien conocida y transmitida desde la Antigüedad y la Edad Media, así como objeto de numerosos comentarios para aclarar su contenido.

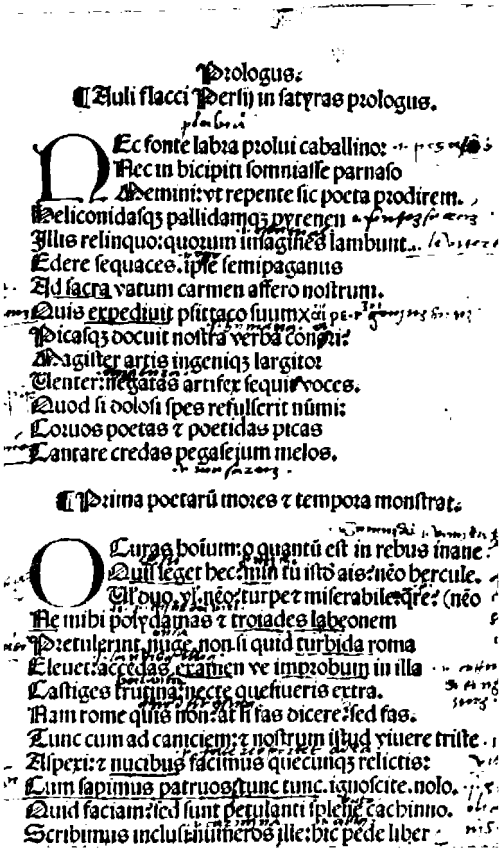
Tras la invención de la imprenta fue un texto impreso múltiples veces en el primer siglo de la imprenta.

3.1.4. Complicaciones de los documentos históricos

El problema con estos documentos, y en concreto, el que se va a utilizar en este proyecto es en cuanto al estado de este, ya que resulta difícil realizar la digitalización debido a determinadas características de estos, entre las que se destacan:

- Diferentes versiones: Uno de los problemas es que para una misma obra surgen diferentes versiones que pueden llegar a diferir bastante entre ellas, debido a errores del impresor o defectos de la imprenta.
- Glosas: Son comentarios manuscritos en los márgenes de un libro, el cual puede significar diferentes motivos, desde explicar el significado del texto en su idioma original a comentarios sobre el mismo. En los impresos más antiguos, en las glosas también se pueden encontrar información sobre el propio texto o variantes al mismo.
- Defectos caracteres: Como se contó anteriormente al tratarse de imprenta se presenta una gran diversidad en los caracteres que componen estos impresos. Se puede suponer que el mismo carácter es exactamente igual en todo el texto, pero no siempre es así, ya que frecuentemente la imprenta se quedaba sin tinta o tenía en exceso, haciendo pequeñas diferencias entre mismos caracteres dentro del documento.

Estos factores contribuyen a imágenes muy irregulares que dan como resultado que los sistemas de OCR convencionales no consigan resultados aceptables con estos documentos antiguos.



Prologus.

¶ Auli flacci Perſij in ſatyras prologus.

Nec fonte labra prolui caballino:
Nec in bicipiti ſomniaſſe parnaſo
Memini: vt repente ſic poeta prodirem.
Heliconidaſq3 pallidamq3 pyrenen
Illis relinquo: quorum imagines lambunt
Edere ſequaces. ipſe ſemipaganus
Ad ſacra vatum carmen affero noſtrum.
Quis expediuit pſittaco ſuum xãi
Picaſq3 docuit noſtra verba conari:
Magiſter artis ingeniq3 largitor
Venter: negatas artifex ſequi voces.
Quod ſi doloſi ſpes refulſerit nũmi:
Coruos poetas & poeticas picas
Cantare credas pegaſejum melos.

¶ Prima poetarũ mores & tempora monſtrat.

O Curas hominum: o quantũ eſt in rebus inane.
Quis legat hec: inquis tu iſtõ ais: nẽo hercule.
Vt quo, vt nẽo: turpe & miſerabile q̃re: (nẽo
Me mihi polydamas & troiades labeonem
Prætulerint nigrę non ſi quid turbida roma
Eleuet: accedas examen ve improbum in illa
Caſtiges trutina: nec quę fueris extra.
Nam rome quis non: at ſi fas dicere: ſed fas.
Tunc cum ad canitiem: & noſtrum iſtud viuere triſte.
Aſperſeris: & nucibus facimus quecunq3 relictis:
Cum ſapimus patruos: tunc tunc. ignoſcite. nolo.
Quid faciam: ſed ſunt petulanti ſplene cachinno.
Scribimus inclusi numeros ille: hic pede liber.

Ilustración 2: Página 1 de Las Sátiras de Persio [5]

3.2. Redes de neuronas

Las redes de neuronas son modelos formados por elementos, llamados neuronas, que se comportan tratando de imitar las funciones básicas de las neuronas humanas. Su principal característica es que dado un conjunto de entrada las redes pueden ajustarse para producir la salida deseada para un determinado problema.

Como se ha dicho las redes están formadas por neuronas que estarán compuestas por una o varias entradas, una función de activación y una salida.

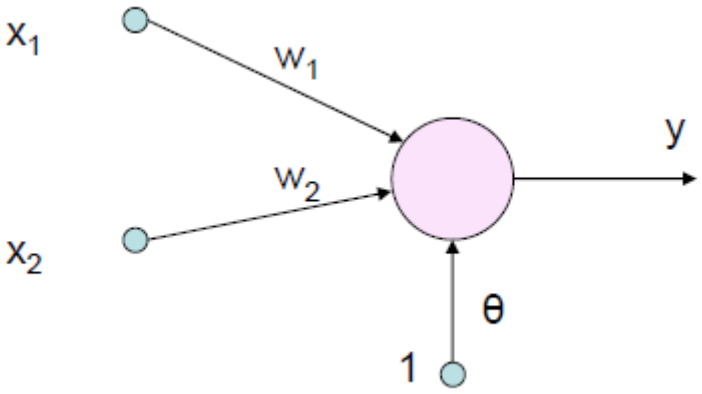
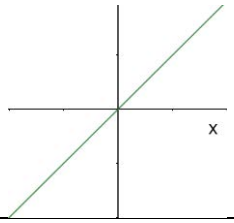
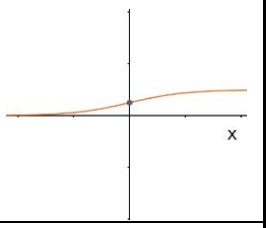
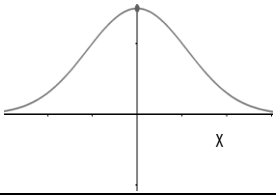


Ilustración 3: Ejemplo neurona artificial

Están caracterizadas por un estado interno que se llama nivel de activación. Este nivel de activación puede ser modificado durante el aprendizaje por la función de activación de la neurona artificial. Se pueden encontrar muchas funciones de activación, pero las más frecuentes son:

Tabla 1: FUNCIONES DE ACTIVACIÓN

	Función	Rango	Gráfica
Lineal	$y = x$	[-∞, + ∞]	
Sigmoidea	$y = \frac{1}{1 + e^{-x}}$	[0,+1] [-1,+1]	
Gaussiana	$y = Ae^{-Bx^2}$	[0,+1]	

Una fase importante de las redes de neuronas es su aprendizaje, en él se trata de entrenar la red neuronal para generar la mejor salida posible para cada dato.

Dicho aprendizaje puede ser de dos tipos:

- Aprendizaje Supervisado: Este tipo de aprendizaje se caracteriza porque el entrenamiento es controlado por un agente externo, que determina la salida que debería generar la red para las entradas determinadas.

  El supervisor obtendrá la salida de la red y la comparará con la deseada. En caso de no coincidir, actualizará los pesos de las entradas con el fin de obtener un resultado óptimo.

- Aprendizaje No Supervisado: Este tipo de aprendizaje se caracteriza porque la salida representa la similitud o semejanza entre la entrada actual y las anteriores. No utiliza información externa, es decir, los datos no contienen información relativa a la salida deseada.

En el proceso del aprendizaje no se utilizan todos los datos disponibles ya que, si se realizara esto, la red podría adaptarse a los datos de entrenamiento y no obtener resultados para datos nuevos, a este problema se le llama sobreajuste. Para evitar esto, los datos se dividirán en diferentes conjuntos en el que cada uno cumplirá una función:

- Conjunto de entrenamiento: Este conjunto es el utilizado en el proceso de aprendizaje y normalmente constituye un 70-80% de los datos. Su función es la de ajustar los pesos de la red, haciendo que se vaya reduciendo el error entre la salida de la red y la salida deseada.
- Conjunto de test: Una vez la red ha sido ajustada, el conjunto de test evalúa la red con un conjunto de datos que no se ha utilizado para ajustar los pesos
- Conjunto de validación: Normalmente se desea encontrar los mejores parámetros de nuestra red a la hora de entrenarla. Para ello se utiliza el conjunto de validación, trata de buscar los mejores parámetros para nuestra red sin tener que usar el conjunto de test para ello, ya que como se ha comentado corre el riesgo de sufrir sobreajuste.

En función de su arquitectura y algoritmo podemos encontrar diferentes tipos de redes de neuronas, que se van a explicar brevemente:

- **Perceptrón simple**

Es una red neuronal básica que genera un hiperplano que permite resolver problemas de clasificación lineal, es decir, permite separar dos tipos de datos linealmente separables.

El aprendizaje de esta red es supervisado, es decir, el entrenamiento es controlado por un agente externo.

Como se puede ver en la ilustración 3, su arquitectura consta únicamente de una capa, además de la de salida y tanto su entrada como su salida utilizan únicamente datos binarios. El algoritmo empleado para obtener la salida por estas redes es, por tanto:

$$y = \begin{cases} 1, & si\ \ w_1x_1 + .. + w_nx_n + U > 0 \\ -1, & si\ \ w_1x_1 + .. + w_nx_n + U \leq 0 \end{cases}$$

Ecuación 1: Salida del perceptrón simple

Donde:

- w_i: Peso i de la neurona
- x_i: Entrada i de la neurona
- U: Umbral

- **Adaline**

Es una red neuronal con una estructura similar al perceptrón simple, la única diferencia notable es la forma de usar la salida. Mientras que en el perceptrón simple la salida era binaria y, por tanto, únicamente tenía en cuenta si se equivocaba o no, en Adaline utiliza la salida real, teniendo en cuenta cuanto se ha equivocado. Teniendo en cuenta esto, esta red permite resolver problemas de regresión lineal. Al igual que el perceptrón simple el aprendizaje es supervisado.

Dicho esto, el algoritmo empleado por esta red es:

$$y = \sum_{i=1}^{n} w_i x_i + U$$

Ecuación 2: Salida de Adaline

En la imagen 4 y 5 se puede ver una comparación de la salida que ofrecería una red de perceptrón simple y una de Adaline respectivamente, y se puede observar que mientras el perceptrón simple trata de clasificar los dos tipos de datos, en el de Adaline trata de reducir el error para cada dato:

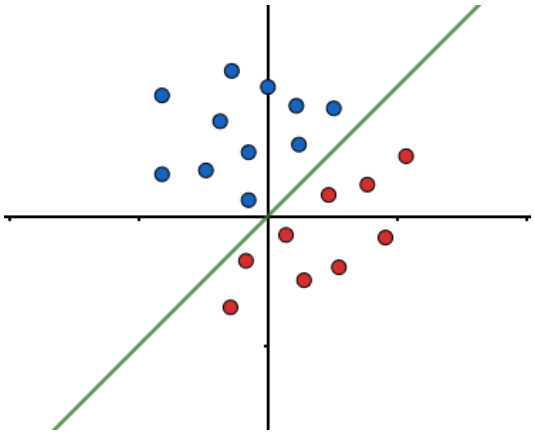


Ilustración 4: Salida perceptrón simple

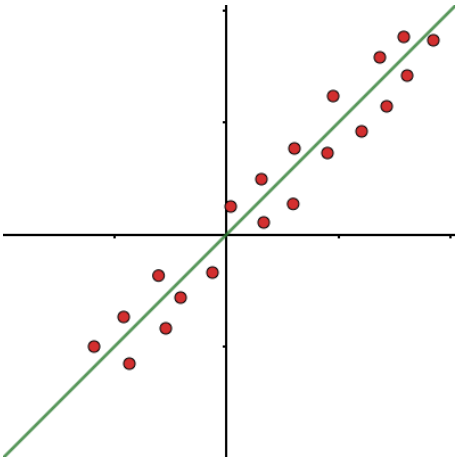


Ilustración 5: Salida perceptrón Adaline

- **Redes de neuronas de base radial**

Esta arquitectura de redes se caracteriza por contar con una única capa oculta la cual se activa en un rango determinado del espacio de entrada. La capa de entrada de la red no lleva pesos asociados a la capa oculta. El proceso de aprendizaje en este tipo de redes puede ser de dos tipos: aprendizaje híbrido o aprendizaje supervisado. En él se ajustan los pesos de los centros, las desviaciones y los pesos de las neuronas.

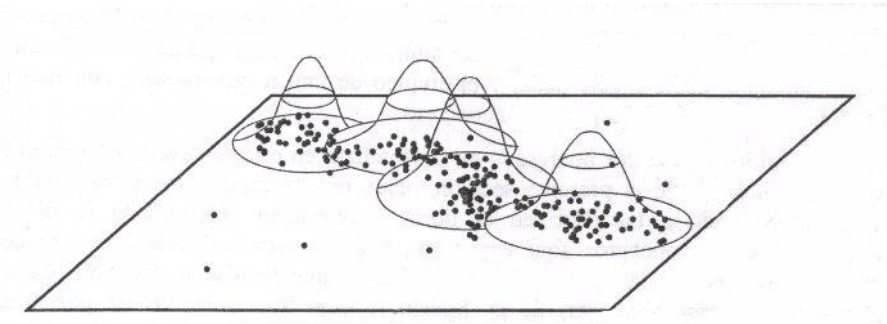


Ilustración 6: Representación red de base radial [6]

La salida de la red vendrá determinada por la fórmula:

$$y_k(n) = \sum_{i=1}^{m} w_{ik}\, \phi i\,(n) + u_k\, para\ k = 1,2,\ldots,r$$

Ecuación 3: Salida de una red neuronal de base radial

Donde:

- w_{ik}: peso de la conexión de la neurona oculta i a la de salida k
- ϕ_i (n): activación de la neurona oculta i
- u_k: umbral de la neurona de salida k

Las redes que se utilizan en el software que emplearemos en el proyecto son el perceptrón multicapa y las recurrentes LSTM, por ello, se van a explicar a continuación con más detalle.

3.2.1. Perceptrón Multicapa

Se puede decir que el perceptrón multicapa es una de las arquitecturas más utilizadas para la resolución de problemas, entre otras cosas por su fácil uso. Esta arquitectura permite resolver problemas de regresión no lineal. Su arquitectura está formada por diferentes capas:

- Capa de entrada: Compuesta por aquellas neuronas en las que se introduce las variables de entrada a la red.
- Capas ocultas: Formada por aquellas neuronas cuyas entradas provienen de capas anteriores y sus salidas pasan a capas posteriores.
- Capa de salida: Neuronas cuyos valores de salida se corresponden con la salida final de la red.

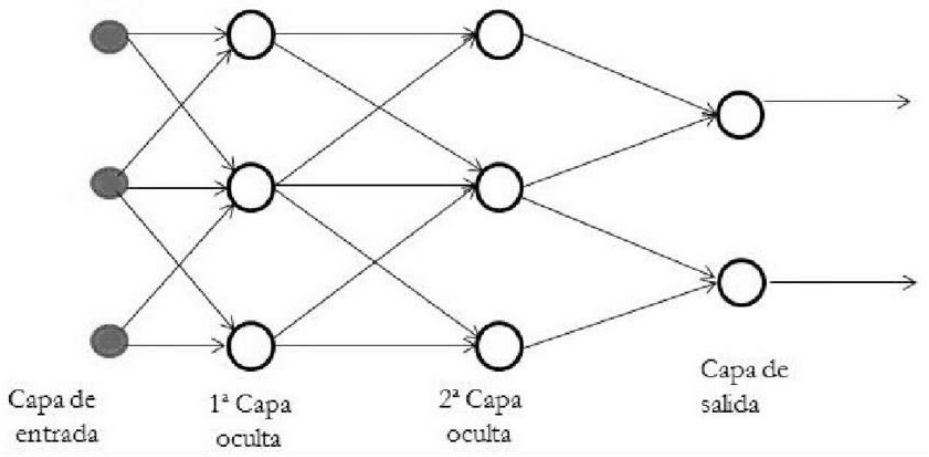


Ilustración 7: Estructura perceptrón multicapa

Esta arquitectura es un aproximador universal, es decir, puede aproximar cualquier función continua gracias a su innovador algoritmo de entrenamiento que surgió de la idea de unir varios perceptrones simples pudiendo resolver problemas no lineales. Su proceso de entrenamiento es el siguiente:

En la fase de aprendizaje se trata de minimizar el error para cada ejemplo de entrenamiento adaptando los pesos y umbrales. Esta función de error se puede definir:

$$E = \frac{1}{N}\sum_{n=1}^{N} e(n)$$

Ecuación 4: Función error proceso de aprendizaje

$$e(n) = \frac{1}{2}\sum_{i=1}^{n_c} (s_i(n) - y_i(n))^2$$

Ecuación 5: Error individual de una entrada n

En este caso E sería la función a minimizar, donde N es el número de datos de entrada, $s_i(n)$ es la salida deseada, $y_i(n)$ es la salida de la red y e (n) es el error para el dato n

En este tipo de redes se suele emplear el método del descenso del gradiente que trata de minimizar la función de error para cada dato de entrenamiento El algoritmo de retropropagación se puede resumir en:

- Para la última capa:

$$w_{ji}^{c-1}(n) = w_{ji}^{c-1}(n-1) + \alpha\delta_i^c(n)a_j^{c-1}(n)$$

Ecuación 6: Calculo del valor del nuevo peso para la última capa

$$u_i^c(n) = u_i^c(n-1) + \alpha\delta_i^c(n)$$

Ecuación 7: Calculo del umbral para la última capa

- Donde:

$$\delta_i^c = (s_i - y_i)y_i(1-y_i)$$

Ecuación 8: Calculo de δ para la última capa

- Para el resto de capas:

$$w_{kj}^c(n) = w_{kj}^c(n-1) + \alpha\,\delta_j^{c+1}(n)a_k^c(n)$$

Ecuación 9: Calculo del valor del nuevo peso para el resto de capas

$$u_j^{c+1}(n) = u_j^{c+1}(n-1) + \alpha\,\delta_j^{c+1}(n)$$

Ecuación 10: Calculo del umbral para el resto de capas

- Donde:

$$\delta_j^{c+1} = a_j^{c+1}(1-a_j^{c+1})\sum_{i=1}^{n_{c+2}}\delta_j^{c+2}\,w_{ji}^{c+1}$$

Ecuación 11: Calculo de δ para el resto de capas

Los pasos para seguir el proceso de entrenamiento aplicando este algoritmo son los siguientes:

1. Se inicializan los pesos y umbrales de las neuronas, normalmente con valores próximos a 0.
2. Se introduce una entrada a la red del conjunto de entrenamiento produciendo la salida de este dato.
3. Se calcula el error cometido por la red para el dato usado (Ecuación 5)

4. Se aplica el algoritmo de retropropagación para obtener los nuevos pesos y umbrales
5. Se repiten los pasos 2,3 y 4 con todos los conjuntos de entrada.
6. Se repiten los pasos del 2 al 5 hasta la parada.

En el proceso de aprendizaje, la hora de elegir el número de iteraciones total puede producir algunos problemas que ya se han mencionado y que van a tratarse de explicar mejor a continuación.

Los modelos generados pueden sufrir dos posibles situaciones que afectarían al estado final del mismo que son: subajuste y sobreajuste

- El subajuste en un modelo ocurre cuando el número de iteraciones que se ha entrenado la red es bajo, y, por tanto, para el conjunto de datos de entrenamiento dicha red no generaliza bien.
- El sobreajuste, por el contrario, es cuando el número de iteraciones es muy alto y se ha generalizado en exceso con el conjunto de entrenamiento. En este caso el error para estos datos será bajo, pero para datos de otros conjuntos será demasiado alto.

Por estos problemas hay que tener en cuenta diferentes criterios a la hora de parar el entrenamiento de la red, sin que sea excesivo ni insuficientes iteraciones. Para ello los criterios normalmente seguidos son:

- El error de entrenamiento permanece estable
- El error de validación permanece estable
- El error de validación empieza a aumentar

3.2.2. Redes de neuronas recurrentes

Las redes recurrentes son redes neuronales que se caracterizan por la utilización de la salida de la red como una entrada más, permitiendo a la red que tenga memoria. En otras palabras, su característica principal es que la información puede persistir introduciendo bucles en el diagrama de la red, es decir pueden recordar cuales eran los estados previos y utilizar esto para determinar el siguiente estado.

Las activaciones de las neuronas ahora no solo dependen de la capa anterior, sino también de la activación de todas las neuronas conectadas a ella, pudiendo ser incluso ella misma.

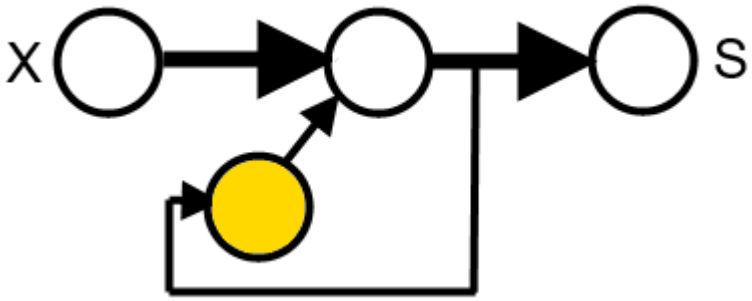


Ilustración 8: Representación red de neuronas recurrente [7]

Al introducir estas conexiones el número de pesos aumenta complicando el aprendizaje con respecto otro tipo de redes. Además, la arquitectura de estas redes de neuronas cuenta con un gran número de capas ocultas.

Podemos encontrar diferentes tipos de redes recurrentes, pero antes de hablar de ellas conviene explicar el término de *Deep Learning*. *Deep Learning* es una técnica de aprendizaje automático que utiliza las redes de neuronas para sus diferentes aplicaciones. Mientras que los métodos más tradicionales son lineales, el *Deep Learning* se encargan de problemas con un nivel de complejidad y abstracción mayor.

Por lo tanto, el *Deep Learning* trata de emular el método de aprendizaje de los humanos para obtener ciertos conocimientos más abstractos y complejos que con los métodos más tradicionales [8]

El problema viene que el algoritmo de retropropagación funciona bien con una capa, pero con muchas no, a esto se le llama el problema del gradiente y para solucionar esto surgieron las redes recurrentes LSTM

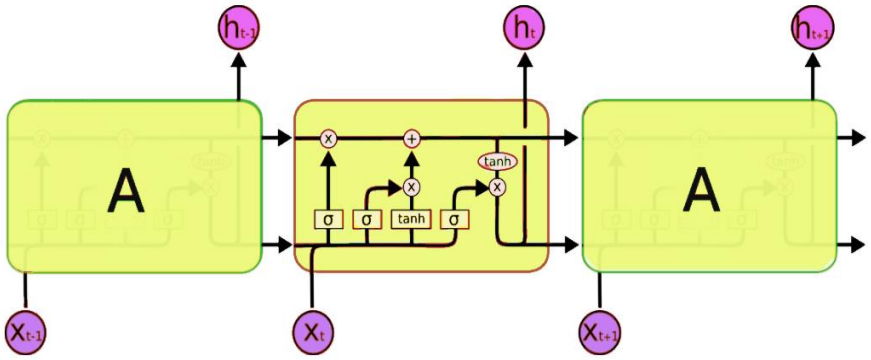


Ilustración 9: Red de neuronas recurrentes LSTM

Las redes recurrentes LSTM se caracterizan porque pueden aprender dependencias largas por lo que recuerdan a más largo plazo. El funcionamiento de estas redes suele ser bastante complicado debido a su compleja arquitectura. En la imagen 9 se puede ver una comparación entre una red neuronal estándar, una red recurrente estándar y una red recurrente LSTM.

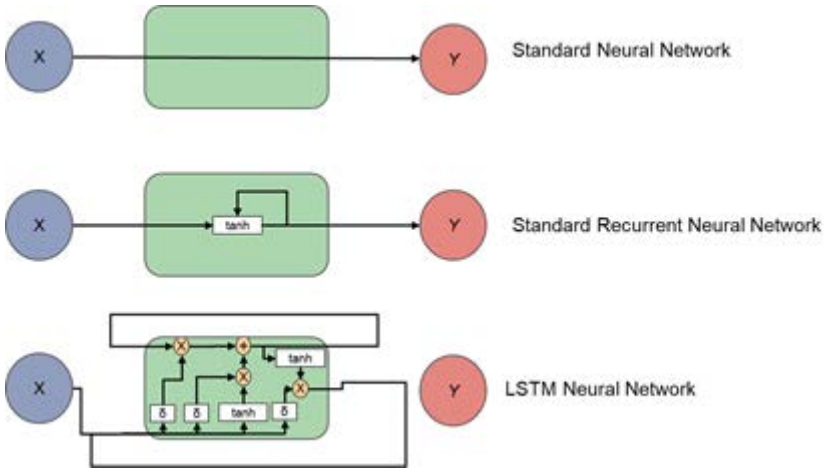


Ilustración 10: Comparativa redes de neuronas

Otro tipo de red recurrente más reciente son las redes GRU's, son un tipo de redes muy similares a las redes LSTM y que cuenta con una arquitectura mucho más sencilla que estas. Las posibles aplicaciones de este tipo de redes son:

- Reconocimiento de voz
- Control de robots

- Traducción de lenguaje de signos

3.2.3. Transfer Learning

Por último, en este punto se va a hablar sobre un tipo de entrenamiento específico, como es el Transfer Learning.

El Transfer Learning es un tipo de aprendizaje caracterizado por la utilización de un modelo previamente entrenado con datos de otro documento. Dicho modelo pre-entrenado se continúa ajustando con los datos del nuevo documento, pero se espera que, o bien el proceso de aprendizaje sea más rápido (menos ciclos a usar), o mejor en error, o requiera de menos líneas del nuevo documento para obtener un modelo adecuado. En otras palabras, se trata de aprovechar la experiencia adquirida por el modelo en un problema distinto (otro documento), para mejorar el entrenamiento del modelo en el nuevo documento.

Es por eso por lo que esta técnica se suele utilizar cuando los datos disponibles son escasos, puesto que se parte de un modelo pre-entrenado con otros datos previos. Los resultados de esta técnica dependen en gran medida en la relación del modelo pre-entrenado con el documento utilizado para el entrenamiento, ya que a más similitudes en cuanto a caracteres e idioma utilizado obtendremos normalmente mejores resultados.

3.3. Digitalización

Entre las aplicaciones de estas redes de neuronas se encuentra la tecnología OCR (Optical Character Recognition) que, como su nombre indica, consiste en el reconocimiento de caracteres. Esta tecnología es aplicable con muy buenas soluciones para textos del siglo XX, pero para los primeros textos escritos no resultan tan eficaces principalmente por los problemas comentados anteriormente.

Con el *Deep Learning*, se empezaron a desarrollar distintas herramientas que facilitaban en gran parte la labor de digitalización. Y aunque este desarrollo sigue en proceso, puesto que aún no se ha llegado al punto óptimo que se desea, se pueden obtener unos resultados aprovechables con alguna de las herramientas disponibles.

A continuación, vamos a ver en profundidad el software que se va a utilizar para este proyecto, así como sus ventajas y desventajas.

3.3.1 Ocropus

Es el software que vamos a utilizar durante el proyecto, es un sistema libre que utiliza Python y se utiliza a través de consola de comandos [9]. Utiliza diferentes arquitecturas como las anteriormente explicadas, perceptrón multicapa y redes recurrentes.

Todos los comandos del software incluyen una variable para mostrar ayuda sobre el propio comando, siendo de gran utilidad a la hora de iniciarse con Ocropus.

Este software no permite únicamente realizar el entrenamiento del modelo, sino que además cuenta con diferentes comandos que nos permite realizar distintas funciones, algunas de ellas interesantes para el proyecto. El orden lógico para generar un modelo en este software es el siguiente:

- Preparación de las imágenes: Se necesitan las imágenes de las páginas con el menor ruido posible y binarizadas a blanco y negro. Este paso se puede realizar en Ocropus, pero no resulta todo lo eficaz que se desearía y en ocasiones puede resultar mejor utilizar algún software externo para realizar este paso.
- Segmentación de páginas: Para realizar el modelo es necesario contar con imágenes en las cuales solo se contiene una línea del documento, para ello se debe segmentar las imágenes de las páginas. El software nos permite realizar este proceso con buenos resultados, aunque como se habló anteriormente, en estos documentos se encuentran varios elementos como son las glosas que complican este trabajo.
- Ground truth: Para todas las líneas que se vayan a utilizar durante el entrenamiento y la evaluación se debe incluir su transcripción en formato txt. El software nos permite generar un html en el que se incluyen todas las imágenes de las líneas junto un cuadrado para introducir dicha transcripción, una vez terminado se puede guardar dicho html y generar los ficheros txt correspondientes como ground truth.
- Entrenamiento: Es el paso principal del proyecto, en él se realiza el entrenamiento de los datos que componen el conjunto de entrenamiento utilizando el ground truth anteriormente incluido. Este proceso incluye bastantes parámetros que pueden ser modificados para optimizar el modelo que genera. En Ocropus el comando a utilizar será ocropus-rtrain y los principales parámetros que se pueden modificar son:
    - NTRAIN: Máximo número de ciclos

- OUTPUT: Nombre con el que se guardarán los modelos.
- SAVEFREQ: Frecuencia en ciclos en las que se guardarán los modelos.
- NTRAIN: Número de ciclos que se entrenarán antes de parar.
- LOAD: Empezar a entrenar a partir de un modelo dado.

En la ilustración 11 se puede ver el modo gráfica que se muestra durante el entrenamiento de cada línea, en el que se puede ver de manera visual la salida que tiene la red en ese momento para cada línea de entrenamiento.

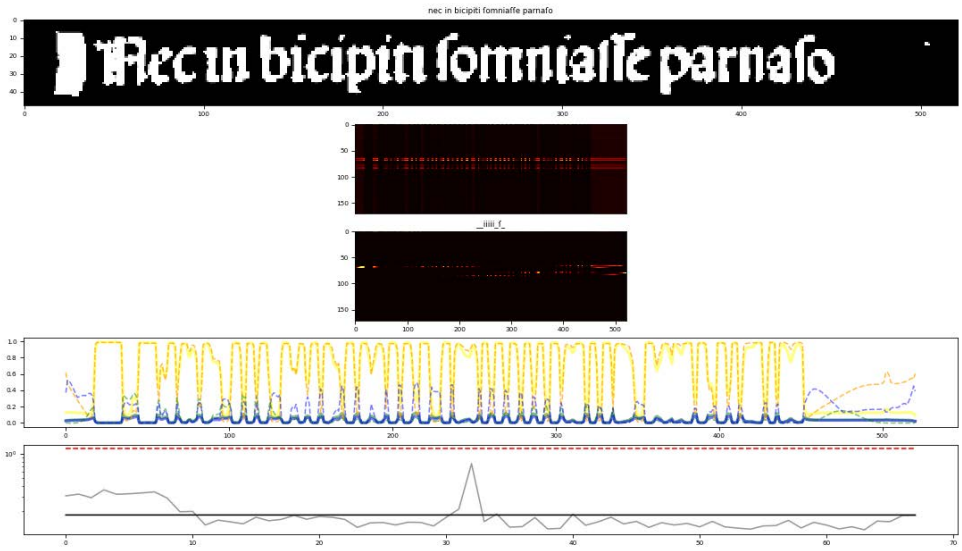


Ilustración 11: Salida gráfica de Ocropus durante el entrenamiento

- Predicción: Una vez se obtiene el modelo podemos pasarle tantas imágenes de líneas como deseemos y él genera en formato txt la salida que ofrece la red entrenada. En Ocropus el comando es ocropus-rpred. Durante esta llamada en la consola se puede ver la salida que tendrá nuestra red sin necesidad de observar los txts generados de manera automática.
- Evaluación: Con la predicción hecha y en los datos que tienen ground truth se puede obtener el porcentaje de error de la red. El comando empleado en Ocropus para esta función será ocropus-errs y como se ha dicho solo se realizará sobre datos que tengan el txt obtenido de la predicción y el ground truth introducido.

En este software se encuentran algunos archivos interesantes que se pueden modificar y que se usaran durante el proyecto de los cuales se van a hablar a continuación:

- Chars.py: Este archivo recoge los diferentes caracteres y glifos que se incluyen en el modelo, por tanto, se deberían incluir todos los caracteres que aparecen en el documento para que puedan ser utilizados a la hora de realizar el entrenamiento del modelo. Es importante resaltar que los glifos se codifican con un código UTF Unicode y por tanto en el reconocimiento de imágenes hay que utilizar siempre el mismo código para los mismos glifos. Este detalle es importante ya que para mismos glifos hay diferentes codificaciones y usarlas puede causar que la red no funcione adecuadamente
- Setup.py: Es el archivo que monta el software y si se desea modificar alguno de los archivos principales del software debe ser ejecutado de nuevo para instalar los cambios realizados, como puede pasar por ejemplo con el antes mencionado Chars.py.

4. EXPERIMENTOS INICIALES

En este capítulo se va a hablar sobre los primeros trabajos que se han realizado en este proyecto. Se describen los datos de partida, el tratamiento de las imágenes y los primeros experimentos realizados.

4.1. Datos de partida

Como se ha mencionado, los datos iniciales consisten en las 22 páginas del libro *Las Sátiras de Persio* impresas en el taller de Arnao Guillén de Brocar en Logroño. Estas páginas contienen alrededor de unas 600 líneas. Para realizar este proyecto se ha contado con las 22 páginas transcritas para que los resultados obtenidos sean más fiables.

4.2. Tratamiento de las imágenes

El primer paso por realizar es el de preparar las imágenes para el entrenamiento. En este caso lo primero que realizaremos será procesar estas imágenes en líneas.

Ocropus contiene una funcionalidad que separa la imagen de cada página en una imagen por cada línea, pero este método cuenta con algunos problemas con documentos antiguos. Contamos con tres problemas principales que son:

- Ruido: en documentos antiguos es muy común encontrar ruido en las imágenes que impiden ver correctamente las distintas líneas de este, por tanto, se debe tratar de limpiar este ruido al máximo para que afecte lo menos posible a la salida de nuestro modelo.

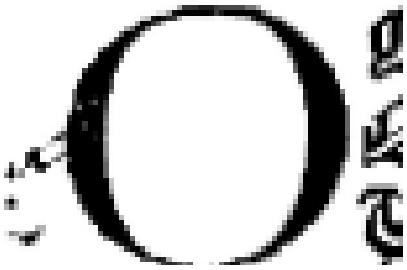


Ilustración 12: Letra capital

- Glosas: en este texto es bastante común el uso de glosas, y este programa no es totalmente efectivo a la hora de diferenciarlas y algunas se incluyen como imagen, mezclándose en la misma imagen procesada, el texto del documento y la glosa En la imagen 13 se puede ver un ejemplo.

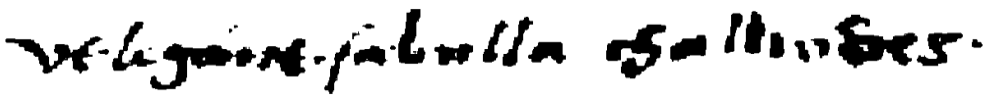


Ilustración 13: Ejemplo glosa en el documento de Persio

- Letras capitales: como se puede ver en la imagen 12 tienen un formato diferente y Ocropus no la detecta como línea dividiendo la letra en diferentes imágenes.

La forma en la que se resolverá estos problemas será en eliminar estas imágenes que no incluyen parte del texto que queremos utilizar, la manera más fácil de realizar esto, será elaborar el ground truth dejando en blanco los huecos de las glosas y líneas en mal estado y eliminar posteriormente todas aquellas imágenes que no tengan el documento de transcripción generado. Otra posible solución hubiera sido simplemente omitirlas, es decir, no eliminándolas, ya que esto no afecta a los resultados que obtendremos, simplemente se eliminaron para una mayor comodidad a la hora de manejarse con los datos.

Ilustración 14: .html generado por Ocropus con las líneas del documento

Una vez tenemos las imágenes que contienen cada una de las líneas, se debe incluir el ground truth para aquellas líneas que vamos a utilizar para entrenar o evaluar. El programa nos permite generar un archivo html en el que cada línea de la imagen 14 contiene por un lado una imagen (de la línea) y un recuadro, donde de manera manual un experto introduce la transcripción con caracteres de texto.

Una vez transcritas las líneas que nos interesan, podemos generar con Ocropus los distintos archivos en formato txt que cumplirán la función de ground truth para las líneas

incluidas su transcripción. Únicamente será necesario generar este fichero para aquellas líneas que utilicemos como entrenamiento o evaluación.

Un ejemplo de cómo se mostraría el ground truth para la línea de la ilustración 14 sería un fichero txt con el siguiente contenido:

“Sede legens celſa, liquido cum plaſmate guttur”

Ilustración 15: Ejemplo línea del documento de Persio

4.3. Primeras pruebas

El objetivo de estas primeras pruebas es obtener una idea de la importancia que tienen los diferentes parámetros que nos permite modificar Ocropus durante el aprendizaje del modelo.

Durante estas pruebas haremos uso únicamente de dos conjuntos de datos, compuestos de líneas del libro de Persio, estos conjuntos son:

- El conjunto de entrenamiento, utilizado para realizar el aprendizaje del modelo, debe contener todos los caracteres que se usan en el libro y lo ideal es que este compuesto del menor número de datos posibles, ya que, como se ha mencionado el esfuerzo de generar el ground truth de los datos es muy costoso, ya que en este tipo de documentos debe ser un experto el que se encargue de transcribir las distintas líneas.
- El conjunto de test, utilizado para comprobar el porcentaje de acierto del modelo. Lo ideal es que contenga todos los caracteres del documento. En estas pruebas este conjunto estará compuesto por todos los datos que no estén en el conjunto de entrenamiento.

Antes de realizar el entrenamiento se debe hacer un último cambio, el programa utiliza un archivo “chars.py” para saber los caracteres, el que viene por defecto no cuenta con todos los caracteres que utiliza esta edición y se deben añadir aquellos caracteres y glifos que no están en este documento. Estos caracteres añadidos fueron:

- ā y el resto de las vocales: Estos caracteres indican que falta una ‘n’ o ‘m’ delante o detrás de la vocal.

- ſ, llamada s longa, es la forma de la s en posición inicial e interior. Solo aparece en minúscula.
- ʒ, este símbolo equivale a varios grupos de caracteres según el contexto. Es una abreviatura a veces de -ue, a veces de -us, a veces de -um.
- ꝑ abreviatura de per

Una vez hecho esto y con los conjuntos de test y entrenamiento con GT ya se podía pasar al proceso de entrenamiento. Aquí la aplicación nos ofrece 3 parámetros alterables. Estos parámetros son:

- Número de ciclos: Iteraciones que se realizan durante el entrenamiento de la red. Además, es interesante el poder guardar y utilizar modelos intermedios, antes de que se alcance el número de ciclos máximo. Durante este primer acercamiento al software se guardó un modelo cada 1000 ciclos hasta los 5000.
- Neuronas ocultas: El número de neuronas controla la complejidad del modelo. El modelo tiene que ser lo suficientemente complejo, pero no demasiado parar que no se produzca sobre-aprendizaje. En Ocropus el valor por defecto es de 100 neuronas ocultas. En estos primeros experimentos se han utilizado como valores 100, 200 y 300 neuronas para evaluar las posibles diferencias con diferentes números de neuronas.
- Razón de aprendizaje: Por defecto 0.0001. Se utilizan como valores 0.0001, 0.001 y 0.005. Inicialmente se utilizó 0.01 pero era un valor demasiado elevado que impedía que el entrenamiento convergiera.

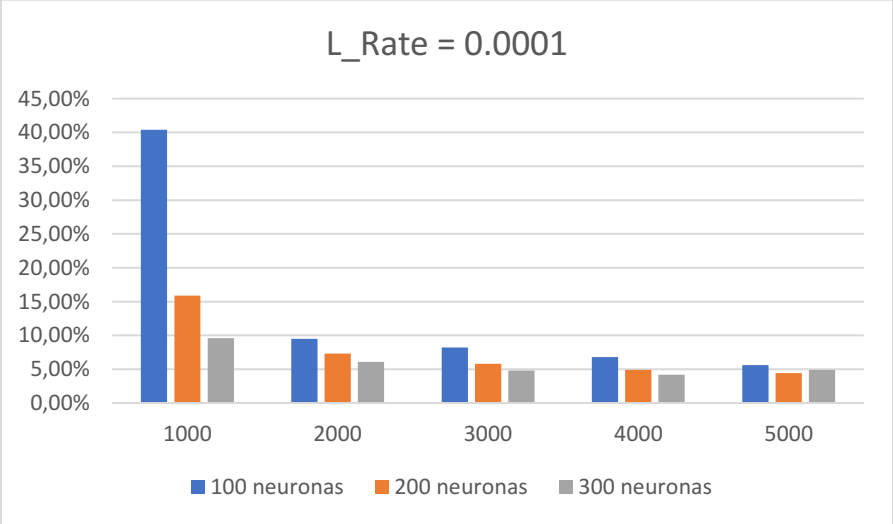
Los resultados obtenidos se pueden ver en la tabla 2.

Tabla 2: PORCENTAJE DE ACIERTO EN EL CONJUNTO DE TEST PRIMERA PRUEBA

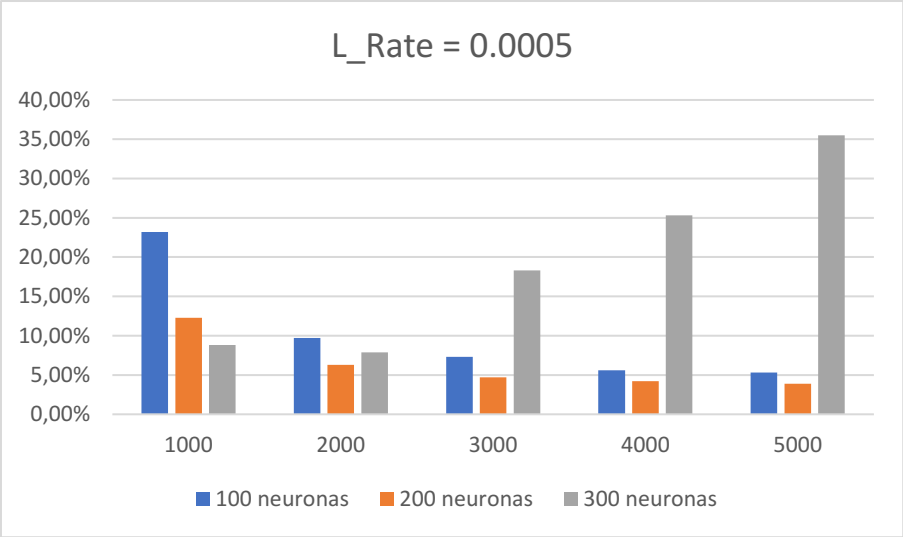
Hiddensize	L_Rate	1000	2000	3000	4000	5000
	0,0001*	40,40%	9,50%	8,20%	6,80%	5,60%
100	0,0005	23,20%	9,70%	7,30%	5,60%	5,30%
	0,001	14,10%	10,40%	5,30%	4,30%	5,20%
	0,0001*	15,90%	7,30%	5,80%	4,90%	4,40%
200	0,0005	12,30%	6,30%	4,70%	4,20%	3,90%
	0,001	10,05%	4,70%	3,90%	3,70%	3,80%
	0,0001*	9,60%	6,10%	4,80%	4,20%	4,90%
300	0,0005	8,80%	7,90%	18,30%	25,30%	35,50%
	0,001	85,20%	85,20%	85,20%	85,20%	85,20%

En la tabla 2 se representa el porcentaje de error del conjunto de test. este porcentaje se determina sobre los caracteres y no sobre las líneas (es decir, es el porcentaje de caracteres correctamente clasificados por la red). En la tabla podemos ver las "hiddensize" que son el número de neuronas ocultas, el l_rate que es la razón de aprendizaje con los valores anteriormente mencionados, y en las columnas siguientes se corresponden al número de ciclos con el que fue entrenado dicho modelo.

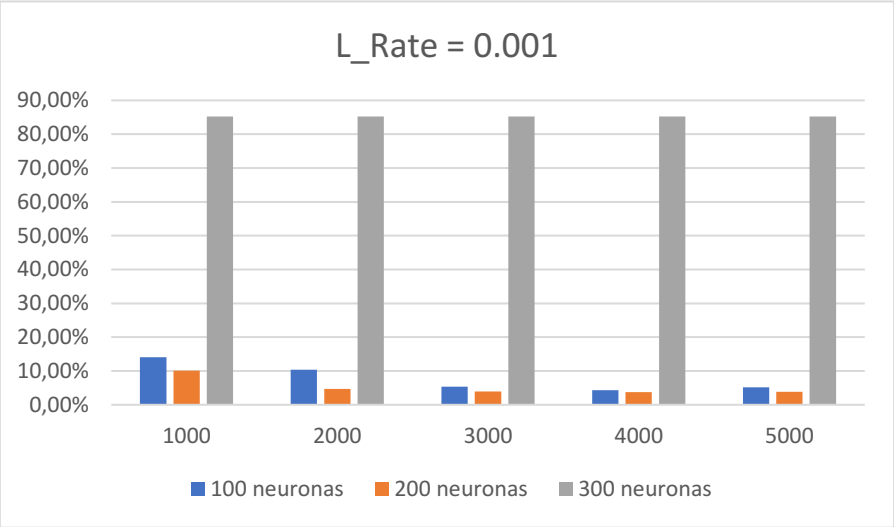
En las gráficas 1,2 y 3 se representa los mismos resultados que en la tabla. En este caso el eje x es el número de ciclos y el eje y el porcentaje de error del conjunto de test. Cada color en las columnas representa un número de neuronas ocultas (100, 200, 300).



Gráfica 1: Error en el conjunto de validación con un factor de aprendizaje de 0.0001



Gráfica 2: Error en el conjunto de validación con un factor de aprendizaje de 0.0005



Gráfica 3: Error en el conjunto de validación con un factor de aprendizaje de 0.001

En la sección 4.4 se analizarán los resultados aquí presentados.

4.4. Análisis de los resultados preliminares

Gracias a estos datos podemos obtener diferentes conclusiones que nos ayudarán para las siguientes pruebas:

- En la tabla 2 podemos observar que cuanto más alto es el nº de neuronas menor debe ser la razón de aprendizaje, ya que el valor L_rate = 0.001 es válido para 100 y 200 neuronas ocultas mientras que para 300 no. Esto se puede ver también muy

claramente en la gráfica 3, donde usar 300 neuronas lleva a obtener errores muy altos. Lo mismo pasa con 0.005 en este caso con 200 neuronas (ver en la gráfica 2).

- En ocasiones una razón de aprendizaje muy alta hace que el modelo tenga errores muy altos. (ver por ejemplo las gráficas 2 y 3).
- No parece que haya un número de neuronas o razón de aprendizaje mejor ya que para todas las opciones probadas obtenemos valores similares.
- El valor de neuronas = 300 parece no ser bueno ya que para funcionar necesita valores de la razón de aprendizaje demasiado bajos. En la gráfica 3 podemos ver este hecho, ya que la razón de aprendizaje alta hace que este modelo no sea funcional para ese número de neuronas.
- El número de ciclos que hemos empleado parece que se queda corto, ya que muchas veces no se termina de estabilizar los resultados. Por ejemplo, en la tabla 2 y las gráficas 1 y 2 se puede ver como para 100 y 200 neuronas y las razones de aprendizaje más pequeñas, parece posible que se pueda mejorar un poco más el error si se continúa más allá de los 5000 ciclos.

Con estas pruebas hemos podido ver cómo los parámetros intervienen en la red, y como los parámetros reaccionan al resultado de la red, esto nos ayudará a poder realizar pruebas más efectivas en los siguientes puntos del proyecto enfocándolos de mejor manera.

5. ESTUDIO EN PROFUNDIDAD

Basándose en los resultados iniciales del capítulo anterior, en este capítulo se procederá a un estudio sistemático del entrenamiento de la red para resolver el problema de digitalización abordado en este trabajo.

5.1. Elección de conjuntos de entrenamiento, validación y test

A continuación, realizaremos un estudio de parámetros utilizando una metodología rigurosa con conjuntos de entrenamiento / validación / test. Para que el conjunto de test sea completamente independiente del proceso de aprendizaje, se utilizará el conjunto de entrenamiento para construir distintos modelos (con distintos parámetros) y el conjunto de validación se usará para compararlos y elegir el mejor.

La idea principal para dividir los datos disponibles en entrenamiento, validación y test es que los 3 conjuntos cuenten con todos los caracteres del documento. Si algún carácter está en entrenamiento y no en test, el conjunto de test no evaluará correctamente al modelo. Y, al contrario, si algún carácter está en test pero no en entrenamiento, difícilmente el modelo aprenderá a clasificar bien dicho carácter. Se empezará tratando de que el conjunto de test contenga todos los caracteres. Para ello se ha desarrollado una script en Python que nos permita saber el número de caracteres usados y poder compararlos con los de todo el documento.

Una vez tengamos un número de páginas en las que se incluyan todos los caracteres, el resto será para el conjunto de entrenamiento y el conjunto de validación. De las cuales 2/3 serán para el conjunto de entrenamiento y el restante de validación, tratando como se ha dicho antes que en cada conjunto haya el máximo número de caracteres distintos.

Con la script ya realizada obtenemos que el documento cuenta con un total de 77 caracteres distintos. Con esto incluimos tantas páginas como sean necesarias empezando desde el final hasta tener un conjunto de páginas con todos los caracteres que formaran el conjunto de test. En este caso se encuentra el problema de que algunos caracteres son poco frecuentes y es imposible contar con 3 conjuntos con todos los caracteres, por eso se tomará como conjunto de test las últimas 7 páginas del documento que incluyen un total de 70 caracteres distintos de los 77 totales.

De las 15 páginas restantes 2/3 de las mismas serán el conjunto de entrenamiento y el restante pertenecerán al de validación. Se procurará que ambos conjuntos tengan el máximo número de caracteres distintos.

Como se ha dicho, el conjunto de validación se utilizará para elegir entre distintas alternativas o parámetros (por ejemplo, el número de neuronas ocultas). Esto se hace así para evitar utilizar el conjunto de test en ningún proceso relacionado con el entrenamiento. El conjunto de test se debe utilizar sólo al final, una vez que ya se hayan entrenado los modelos y elegido los parámetros. Sin embargo, siempre que se evalúe un modelo con los datos de validación, también se evaluará con el de test, para comprobar si ocurren resultados coherentes.

5.2. Pruebas definitivas

Antes de empezar con los entrenamientos, hay que hablar sobre los valores para los parámetros que utilizaremos. Gracias a las primeras pruebas realizadas tenemos una visión más completa sobre qué valores utilizar:

- Número de ciclos: El número de ciclos utilizado se quedó muy corto ya que había casos en los que el error no había aun empezado a estabilizarse. Para ello se decidió que en estas pruebas se ampliara a 8500 ciclos, además de ir guardando cada modelo cada 500 ciclos en vez de cada 1000.
- Neuronas ocultas: En las anteriores pruebas vimos que 300 neuronas era un valor demasiado alto, que resultaba en errores muy altos en algunas situaciones. Por tanto, se decidió que en vez de realizar la prueba con 300 neuronas se hiciera con 50.
- Razón de aprendizaje: En Ocropus encontramos por defecto 0.0001 como valor para la razón de aprendizaje. Se utilizan como valores 0.0001, 0.001 y 0.005. Inicialmente se utilizó 0.01 pero era un valor demasiado elevado que daba malos resultados.

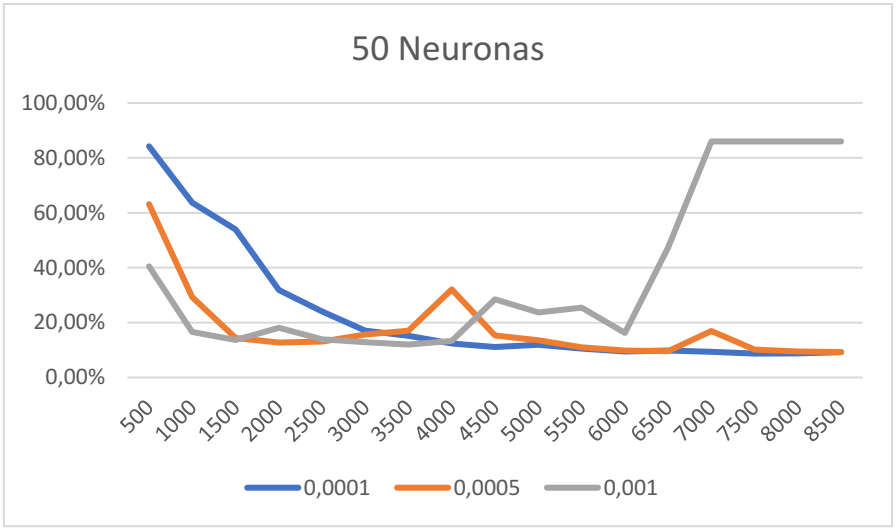
En la tabla 3 se puede ver el porcentaje de error para el conjunto de validación, mientras que los resultados integros del conjunto de test se encuentran en el Anexo A.

Tabla 3: PORCENTAJE DE ERROR DE LOS MODELOS EN EL CONJUNTO VALIDACIÓN

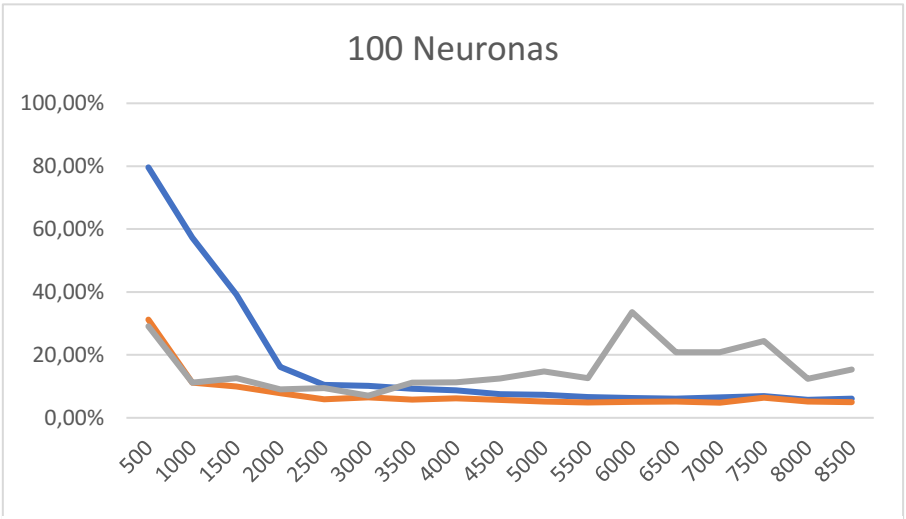
		500	1000	1500	2000	2500	3000	3500	4000	4500	5000	5500	6000	6500	7000	7500	8000	8500
50	0.0001*	84,21%	63,70%	53,91%	31,82%	24,07%	17,09%	15,16%	12,42%	11,12%	11,92%	10,54%	9,49%	9,84%	9,31%	8,70%	8,79%	9,20%
	0.0005	63,14%	29,26%	14,34%	12,71%	13,08%	15,66%	17,08%	32,11%	15,32%	13,59%	10,99%	9,83%	9,59%	16,97%	10,11%	9,43%	9,27%
	0.001	40,51%	16,59%	13,69%	18,06%	13,91%	12,91%	11,99%	13,34%	28,49%	23,64%	25,49%	16,22%	47,41%	--	--	--	--
100	0.0001*	79,63%	57,33%	39,18%	16,16%	10,51%	10,17%	9,22%	8,79%	7,48%	7,30%	6,62%	6,27%	6,14%	6,50%	6,87%	5,73%	6,12%
	0.0005	31,23%	11,19%	9,95%	7,86%	5,91%	6,52%	5,82%	6,16%	5,73%	5,14%	4,90%	5,08%	5,17%	4,82%	6,37%	5,17%	4,94%
	0.001	29,14%	11,24%	12,60%	9,09%	9,45%	7,05%	11,24%	11,33%	12,48%	14,78%	12,62%	33,61%	20,85%	20,91%	24,38%	12,42%	15,32%
200	0.0001*	64,65%	21,55%	11,38%	8,86%	7,88%	6,61%	6,66%	5,67%	5,05%	5,33%	5,41%	5,07%	4,89%	4,92%	5,50%	4,64%	5,42%
	0.0005	28,44%	12,06%	8,43%	6,09%	5,84%	5,41%	5,64%	6,07%	5,91%	4,83%	4,85%	5,19%	4,76%	4,65%	4,96%	5,89%	9,72%
	0.001	18,04%	9,76%	8,27%	6,68%	19,96%	36,05%	22,19%	--	--	--	--	--	--	--	--	--	--

En la tabla 3 la primera columna indicará el número de neuronas, la segunda la razón de aprendizaje y las siguientes columnas el número de iteraciones que ha sido entrenado ese modelo.

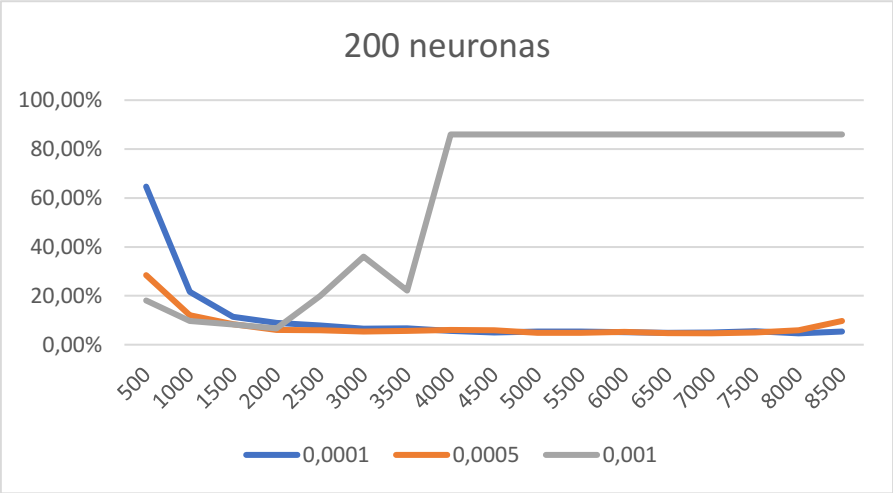
Debido a las grandes dimensiones de la tabla, se han desarrollado las gráficas 4,5 6 para observar y analizar con mayor facilidad estos datos de la tabla 3. En estas gráficas podemos ver la evolución del error en el conjunto de validación con 50,100 y 200 neuronas, respectivamente. En cada gráfica encontramos 3 líneas que representan los tres valores de la razón de aprendizaje usados.



Gráfica 4: Porcentaje de error en el conjunto de validación para 50 neuronas



Gráfica 5: Porcentaje de error en el conjunto de validación para 100 neuronas



Gráfica 6: Porcentaje de error en el conjunto de validación para 200 neuronas

En estas gráficas podemos ver algunos detalles generales:

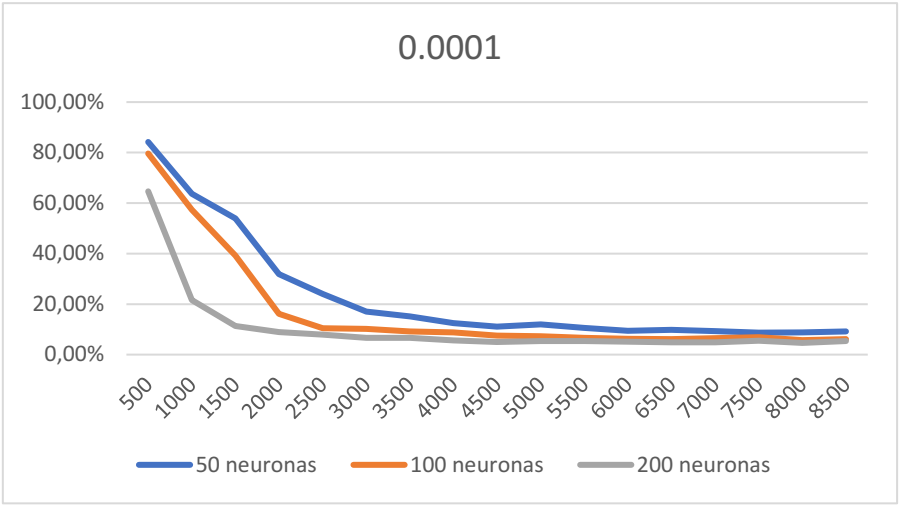
- Parece que 8000 ciclos son excesivos para valores con razón de aprendizaje de 0.001, ya que llega a su punto óptimo mucho antes
- Como se puede ver en la tabla 3 en muchos de los modelos se puede llegar a alcanzar un porcentaje de error de alrededor un 5%
- Se puede ver cómo para una razón de aprendizaje más baja, más tarde se llega a los valores óptimos del modelo, pero se llega de manera más gradual, a errores más pequeños, y sin que en general, ocurra que el error comience a empeorar.

Además, podemos observar también detalles en cada gráfica por separado, dependiendo del número de neuronas y su comportamiento sobre el modelo.

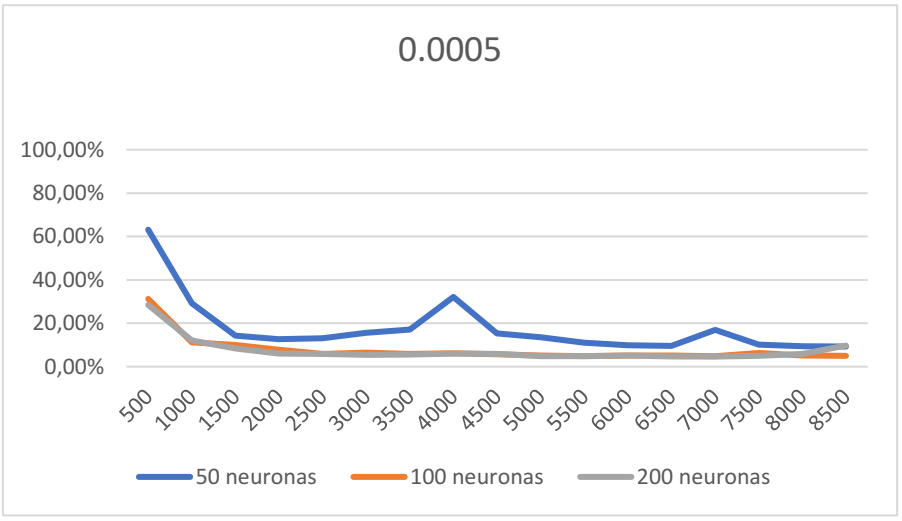
50 neuronas no parece ser un valor bueno para nuestros modelos, ya que el porcentaje de error más bajo obtenido (8,70%) es más alto que el menor total obtenido (4,64%)

Para 100 neuronas, que recordemos es el valor por defecto en Ocropus, se obtienen unos buenos resultados, sobre todo para una razón de aprendizaje 0.0005 llegando a encontrarse entre los 3 mejores.

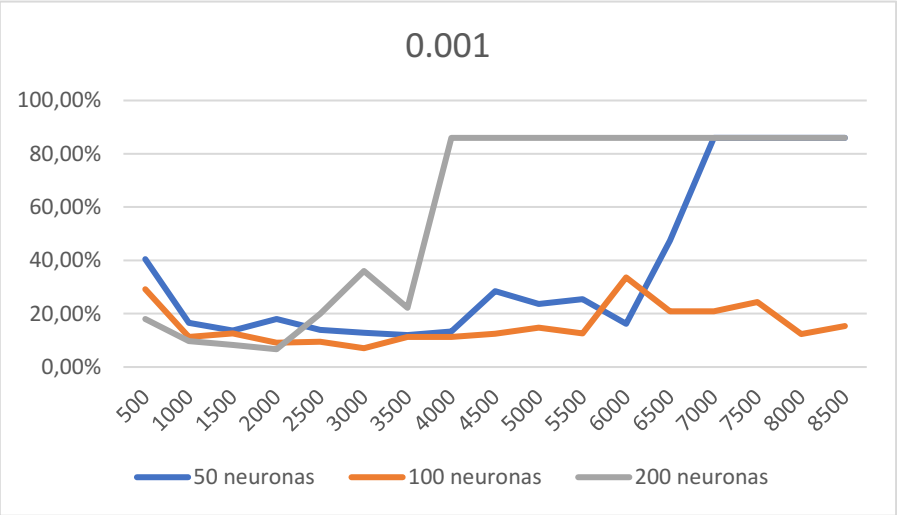
Para 200 neuronas encontramos los 2 mejores valores obtenidos, y aunque es cierto, que el entrenamiento tarda más que con 100 neuronas, este tiempo no es tan grande como para despreciar estos resultados.



Gráfica7: Porcentaje de error en el conjunto de validación para factor de aprendizaje igual a 0.0001



Gráfica 8: Porcentaje de error en el conjunto de validación para factor de aprendizaje igual a 0.0005



Gráfica 9: Porcentaje de error en el conjunto de validación para factor de aprendizaje igual a 0.001

En las gráficas 7,8 y 9 se observarán los mismos resultados, pero mostrándolos por razón de aprendizaje, ya que en ellos podemos observar directamente comparaciones entre distintos números de neuronas.

En la gráfica número 7 correspondiente a la razón de aprendizaje más pequeña, se observa un descenso muy gradual de todos los modelos. Se puede ver que, de los tres números de neuronas, 50 es el valor menos adecuado. Y también, podemos entender porque Ocropus otorga ese valor como razón de aprendizaje por defecto, ya que para diversos números de neuronas obtenemos un comportamiento muy similar para los modelos. Se puede entender que la razón de aprendizaje bajo hace que el entrenamiento del modelo sea muy lento.

En la gráfica 8 podemos ver que, en la tasa de aprendizaje intermedia, el valor de50 neuronas da lugar a un comportamiento inestable, mientras que no se observan diferencias entre 100 y 200 neuronas.

Y por último en la gráfica 7 se confirma que el valor de 0.001 para razón de aprendizaje es demasiado alto para los modelos, ya que da lugar a comportamientos inestables, con tendencia a incrementar el error pasado cierto número de ciclos. Además, como se puede ver en la tabla 3 ninguno de los tres modelos generados con esta razón de aprendizaje otorga un porcentaje de error bajo, ya que el mínimo conseguido de estos tres (6.68%) se

encuentra lejos del mínimo conseguido con todos los modelos (se puede ver también comparando con las gráficas 7 y 9).

En resumen, con estas pruebas hemos obtenido mucha información sobre el funcionamiento del software y cómo se comporta la red para los distintos parámetros. En este caso se puede decir que 50 neuronas no es un valor adecuado en comparación con los otros valores usados, y a su vez 0.001 también es demasiado alto para estos modelos.

5.3. Escoger los parámetros óptimos

En esta sección se va a tratar de elegir el mejor modelo basándonos en los errores de validación concretos, que eran difíciles de visualizar en las gráficas, por lo que se utilizarán tablas. Se ha elaborado una tabla resumida con los mejores modelos para cada conjunto de valores.

En la tabla 4 se pueden ver todos los errores de validación para todas las combinaciones de parámetros (neuronas ocultas, razón de aprendizaje y número de ciclos). En ella la columna ciclos nos indica cuando se alcanza el error mínimo en validación.

Se puede observar que los resultados con 50 neuronas son bastante peores en general (como ya se apreciaba en las gráficas). Los mejores resultados se obtienen con 200 neuronas (entre el 4,6% y el 6,7%), aunque no son mucho mejores que los obtenidos con 100 neuronas (entre el 4,8% y el 7%). En ambos casos (100 y 200 neuronas), la razón de aprendizaje de 0.001 es demasiado grande, y con un valor de 0.0005 se obtienen buenos resultados en ambos casos, además de obtener el óptimo en un número razonable de ciclos (7000 ciclos con 0.0005 vs. 8000 ciclos con 0.0001).
Como se ha dicho al principio de este capítulo, los resultados se elegirían sobre los resultados del conjunto de validación, pero también se iban a analizar los datos del conjunto de test, y sobre todo parece interesante comparar si los mejores modelos obtenidos en ambos conjuntos comparten los mismos valores de sus parámetros.

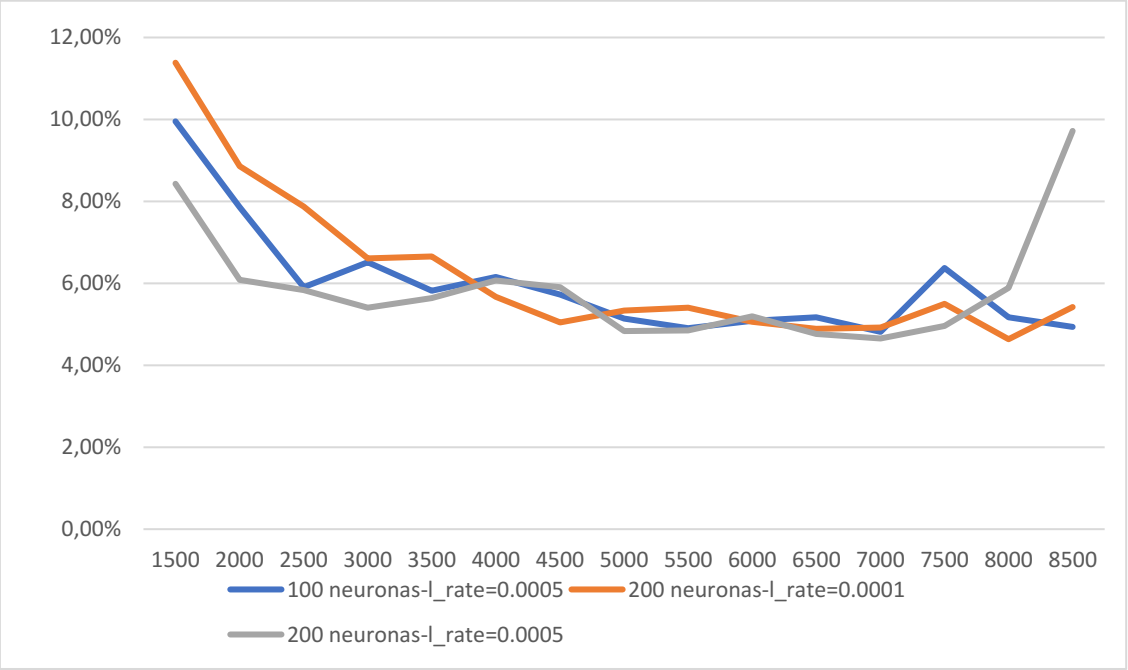
En la tabla 4 también encontramos una columna con el error obtenido para dichos parámetros.

Tabla 4: PORCENTAJE DE ERROR POR MODELOS

Neuronas	L_Rate	Ciclos	%ErrorVali	%ErrorTest
	0.0001*	7500	8,70%	3,67%
50	0.0005	8500	9,27%	4,52%
	0.001	3500	11,99%	5,83%
	0.0001*	8000	5,73%	2,17%
100	0.0005	7000	4,82%	1,91%
	0.001	3000	7,05%	3,04%
	0.0001*	8000	4,64%	1,61%
200	0.0005	7000	4,65%	1,34%
	0.001	2000	6,68%	2,35%

Se puede observar que se cumple una relación entre los resultados obtenidos para el mismo conjunto de valores para los parámetros. Este hecho se puede ver en que en ambos errores se mantiene la relación de los tres mejores conjuntos de parámetros.

Como se ha visto en la tabla 4, hay 3 modelos que cuyo error de validación está por debajo del 5%, por lo tanto, se analizarán de forma conjunta en la gráfica 10.



Gráfica 10: Comparación del error de validación entre los 3 mejores modelos

En este caso partimos del ciclo 1500 para ver mejor la evolución de los modelos a lo largo de los ciclos, en ella podemos observar que existe una clara igualdad entre los 3 modelos, sin haber notables diferencias entre ellos más que su inicio y final.

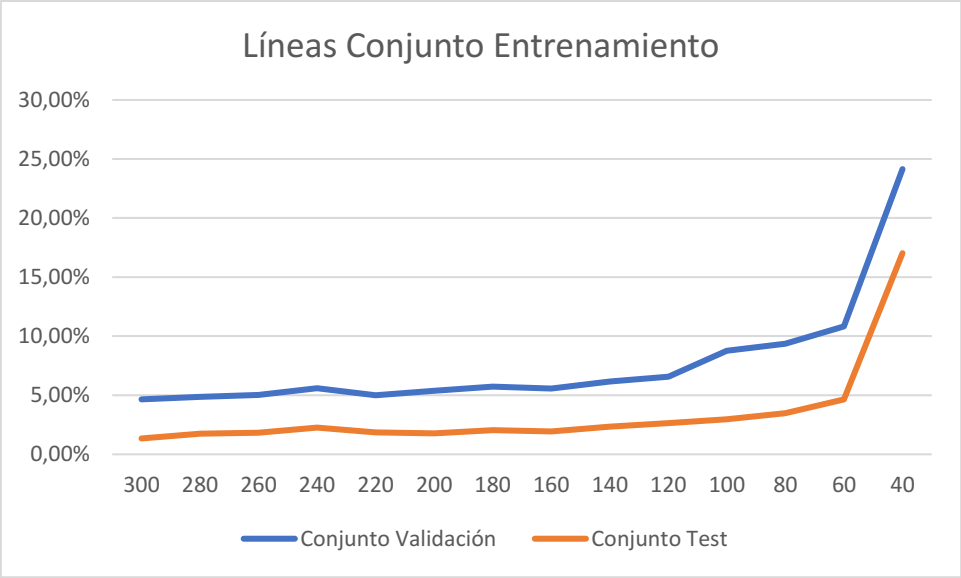
En cualquier caso, y para seguir un criterio sistemático, en los siguientes puntos de este proyecto utilizaremos como mejor modelo aquel que obtiene el menor error de validación en la tabla 4. En este caso 200 neuronas y 0.0001 como razón de aprendizaje.

5.4. Determinar la relación entre tamaño de conjunto de entrenamiento y porcentaje de error

Una vez escogidos los parámetros óptimos, el siguiente objetivo es realizar un estudio acerca de cuál es la cantidad mínima de datos de entrenamiento para obtener buenos resultados. A pesar de que cuantos más datos de entrenamiento se usen, mejores serán los resultados, esto tiene un coste computacional, por un lado, y en general, un coste en la generación de datos de aprendizaje, puesto que tienen que ser transcritos manualmente por un experto. Por ello, es conveniente conocer el tamaño mínimo de la muestra necesario para alcanzar un buen error.

Retomando los conjuntos utilizados en las pruebas anteriores, el conjunto de entrenamiento consistía en un 60% del total de datos, y el objetivo es comprobar que ocurre con el error si se usa un tamaño menor. Este 60% consiste aproximadamente de un total de 300 datos o líneas del documento, los cuales iremos reduciendo de 20 en 20 para ver cómo afecta esto al modelo (esto es lo que habitualmente se conoce como curva de aprendizaje).

En la gráfica 11 podemos ver el comportamiento del modelo con el número de datos en el conjunto de entrenamiento. Podemos ver que, aunque es lógico que vaya empeorando a medida que quitamos datos de entrenamiento, el gran salto no llega hasta que sólo se usan 40 líneas dando valores notablemente peores (se pasa de un 5% de error en test a cerca de un 17%). También podemos decir que a partir de las 140 o 120 líneas el modelo también empeora de forma notable, aunque no de la misma manera, ya que pasa de un 6,57% a 8,76%.



Gráfica 11: Porcentaje de error del modelo según el número de datos en el conjunto de entrenamiento

Como conclusión interesante de este estudio, parece que al menos para este documento es suficiente con la mitad de las líneas que se venían usando (unas 160), para obtener unos resultados buenos, muy similares a los obtenidos con todas las líneas. Esta conclusión seguramente sea extensible a textos similares y proporciona información práctica a los transcriptores humanos, puesto que, en principio, basta con que transcriban alrededor de 160 líneas de un nuevo documento, entrenen el modelo y lo usen para transcribir el resto de líneas del documento. Dado que se espera que el modelo ya tenga una tasa de errores baja, similar a lo observado en este apartado, a los transcriptores ya sólo les queda la tarea de corrección, menos costosa que la transcripción directa.

6. TRANSFER LEARNING

En este capítulo se va a hablar sobre la técnica de transfer learning, tanto de la definición de este como de la prueba realizada sobre el documento de Persio, así como el análisis de los resultados obtenidos del mismo.

6.1. Definición

Como se ha mencionado en la sección 2 el transfer learning es un tipo de aprendizaje caracterizado por la utilización de un modelo previamente entrenado con datos de otro documento. En este punto se va a tratar de relacionar este concepto con el software utilizado Ocropus.

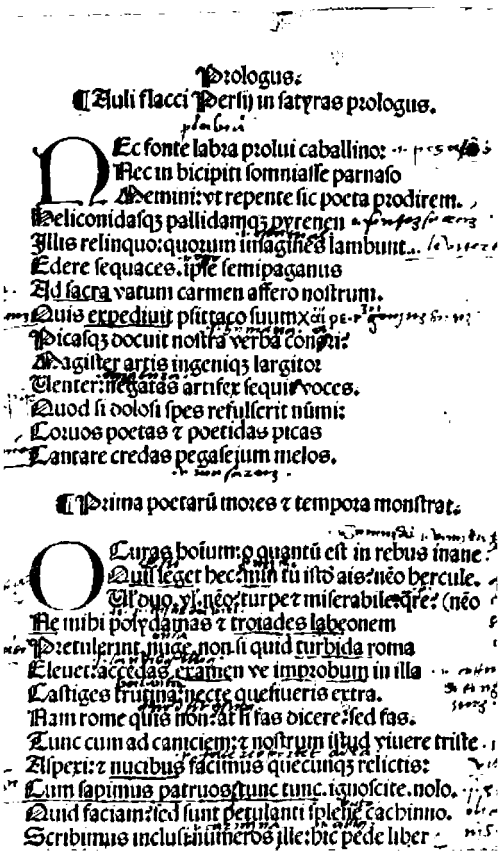
En el artículo [10], se puede ver un ejemplo de esta técnica aplicada en Ocropus con 7 libros distintos y unos resultados bastante prometedores en los que se puede observar que con el modelo pre-entrenado se obtienen resultados buenos mucho antes.

En esta sección, se va a realizar un estudio similar de digitalización de un nuevo documento, en este caso la edición de *Tópica* de Cicerón impresa en el año 1506 en Logroño [11]. El objetivo es comparar los resultados que obtenemos realizando el mismo tipo de entrenamiento que en puntos anteriores con otro, utilizando un modelo de Persio previamente entrenado para partir de él (de hecho, se trata del modelo que ya entrenamos en la anterior sección).

Esta edición de Cicerón se realizó también en el taller de Arnao Guillen de Brocar en una época parecida, en este caso en el 1506.Se ha elegido este documento tanto por la época, similar a la del documento de Persio, como por el taller, ya que en esta época se pueden encontrar grandes diferencias entre diferentes talleres, y como se ha mencionado antes a mayor similitud entre los documentos mejor será a priori el resultado.

La gran diferencia entre estos documentos reside en la tipografía ya que en el documento de Persio nos encontramos tipos góticos, en la edición de Cicerón se utiliza tipos redondos romanos con, incluso, alguna palabra en griego.

En las ilustraciones 16 y 17 podemos ver una página para cada documento utilizado de manera que se pueden ver las diferencias entre estos de una manera más clara.



Prologus.

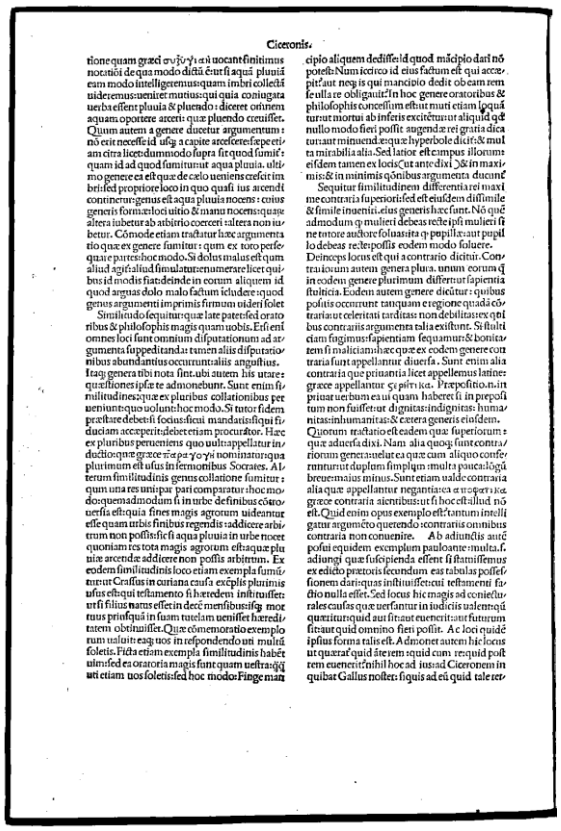
Auli flacci Persij in satyras prologus.

Nec fonte labra prolui caballino:
Nec in bicipiti somniasse parnaso
Demum: vt repente sic poeta prodirem.
Heliconidasq; pallidamq; pyrenen
Illis relinquo: quorum imagines lambunt
Hedere sequaces. ipse semipaganus
Ad sacra vatum carmen affero nostrum.
Quis expedivit psittaco suum χαῖρε
Picasq; docuit nostra verba conari:
Magister artis ingeniiq; largitor
Venter: negatas artifex sequi voces.
Quod si dolosi spes refulserit nūmi:
Corvos poetas & poeticas picas
Cantare credas pegaseium melos.

Prima poetarū mores & tempora monstrat.

O Curas hominum: o quantū est in rebus inane?
Quis legat hec: nisi tu isto ais: nēo hercule.
Vt ovo. vt nēo: turpe & miserabile q̄re? (nēo
Ne mihi polydamas & troiades labeonem
Pretulerint iuuage. non si quid turbida roma
Eleuet: accedas examen ve improbum in illa
Castiges trutina: nec te quesiueris extra.
Nam rome quis non: at si fas dicere: sed fas.
Tunc cum ad canitiem: & nostrum istud viuere triste.
Aspexi: & nucibus facimus quecunq; relictis:
Cum sapimus patruos: tunc tunc. ignoscite. nolo.
Quid faciam: sed sunt petulanti splenis cachinno.
Scribimus inclusi: numeros ille: hic pede liber.

Ilustración 16: Página del libro Las Sátiras de Persio



Ciceronis.

tione quam græci συζυγίαν uocant finitimus notationi de qua modo dictū erat si aquā pluuiā eam modo intelligemus quam imbri collectā uideremus uenire: murus qui aquā contagiat uerba essent pluuia & pluendo: dicere oportere aquam oportere arceri: quæ pluendo ereuisset. Quum autem a genere ducetur argumentum: nō erit necesse id usq; a capite arcesere: sæpe enim citra licet eadem modo supra sit quod sumit: quam id ad quod sumitur aut aqua pluuia. ultimo genere ei est quæ de cælo ueniens crescit in breuiſed propriore loco in quo quasi ius arcendi continetur: genus est aquæ pluuiæ nocens: cuius generis formæ loci uitio & manu nocens aquæ altera subetur ab arbitrio coerceri altera non iubetur. Cōmode etiam tractantur hæc argumentatio quæ ex genere sumitur: quem ex toto perfequare partes: hoc modo. Si dolus malus est quem aliud agit aliud simulatur enumerare licet quibus id modis fiat deinde in eorum aliquem id quod arguas dolo malo factum esse dicere: quod genus argumenti in primis firmum uideri solet

Simili modo sequitur quæ late patet: sed oratoribus & philosophis magis quam uobis. Etsi enim omnes loci sunt omnium disputationum ad argumenta suppeditanda: tamen aliis disputationibus abundantius occurrunt aliis angustius. Itaq; genera tibi nota sint: ubi autem his uteare: quæstiones ipsæ te admonebunt. Sunt enim similitudines: quæ ex pluribus collationibus perueniunt: quas uolunt hoc modo. Si tutor fidem præstare debet: si socius: sicui mandatis: si qui fiduciam acceperit debet etiam procurator. Hæc ex pluribus perueniens quo uult appellatur inductio: quæ græce παραγωγή nominatur: qua plurimum est usus in sermonibus Socrates. Alterum simillitudinis genus collatione sumitur: quum una res uni par comparatur: hoc modo sequenda modum si in urbe definibus cōtrouersia est: quia fines magis agrorum uidentur esse quam urbis finibus regendis: addicere arbitrum non possis: si aqua pluuia in urbe noceat quoniam res tota magis agrorum est: aqua pluuia arcenda: addicere non possis arbitrum. Ex eodem simillitudinis loco etiam exempla sumuntur. Crassus in curiana causa explicat plurimis usus est: qui testamento si heredem instituisset: si filius natus esset in decē mensibus: si q̄ mortuus priusquā in suam tutelam uenisset hereditatem obtinuisset. Quæ cōmemoratio exemplorum usus est: nos in respondendo uim multi soletis. Ficta etiam exempla simillitudinis habet uim: sed ea oratoris magis sunt quam uestra: q̄q; uti etiam uos soletis: sed hoc modo: Finge mancipio aliquem dedisse: id quod mancipio dari nō potest: Num ideirco id eius factum est: qui accepit? aut neq; is qui mancipio dedit ob eam rem se ulla re obligauit? In hoc genere oratoribus & philosophis concessum est: ut muti etiam loquantur: ut mortui ab inferis excitentur: ut aliquid q̄d nullo modo fieri possit augendæ rei gratia dicatur: ut minuendæ quæ hyperbole dicit: & multa mirabilia alia. Sed latior est eorum illorum: isdem tamen ex locis: ut ante dixi. Sic in maximis: & in minimis cōmibus argumenta ducuntur.

Sequitur simillitudinem differentia rei maximæ contraria superiori: sed et eiusdem dissimilis & simile in genere. eius generis hæc sunt. Nō quæ admodum q̄ mulieri debeas recte ipsi mulieri fieri nō recte autore foluaris: q̄ pupillæ: aut pupillo debeas recte ipsi possis eodem modo soluere.

Deinceps locus est qui a contrario dicitur. Contrariorum autem genera plura. unum eorum q̄ in eodem genere plurimum differunt: sapientia stultitia. Eodem autem genere dicitur: quibus positis ocurrunt tanquam e regione: quæ sint contraria: ut celeritati tarditas: non debilitas: nec quibus contraria argumenta talia existunt. Si stultiam fugimus: sapientiam sequamur: & bonitatem si malitiam: hæc quæ ex eodem genere contraria sunt appellantur diuersa. Sunt enim alia contraria quæ priuantia licet appellemus latine: græce appellantur κατὰ στέρησιν. Præpositionem in priuat uerbum ea ui quam haberet si in præpositum non fuisset: ut dignitas: indignitas: humanitas: inhumanitas: & cetera generis eiusdem.

Quorum tractatio est eadem quæ superiorum: quæ aduersa dixi. Nam alia quoq; sunt contrariorum genera: uelut ea quæ cum aliquo conferuntur: ut duplum: simplum: multa: paucalogū breue: magnus: minus. Sunt etiam ualde contraria alia quæ appellantur negantiaea. enquem modum græce contraria: nisi ibus: ut si hoc est illud. nō est. Quid enim opus exemplo est? tantum intelligatur argumēto quærendo: contraria omnibus contraria non conuenire. Ab adiunctis autē possit equidem exemplum pauloante multa. f. adiungi quæ fuerint pienda essent si tam isse mus ex edicto prætoris secundum eas tabulas possessionem dari: quas instituisset: cui testamenti factio nulla esset. Sed locus hic magis ad coniecturales causas quæ uersantur in iudiciis ualet: nēq; quæritur: quid aut fit: aut euenerit: aut futurum sit: aut quid omnino fieri possit. Ac loci quidē ipsius forma talis est. Admonet autem hic locus ut quæras quid ante rem: quid cum re: quid post rem euenerit: nihil hoc ad ius: ad Ciceronem in quibus Gallus noster: figuis ad eū quid tale ter-

Ilustración 17: Página del libro Tópica de Ciceron

Aparte de las diferencias a simple vista de la distribución de páginas entre ambos documentos, hay pequeñas diferencias en caracteres específicos que en esas imágenes no pueden ser observados correctamente, para ello se pueden observar las ilustraciones 18,19,20 y 21 que contienen los mismos caracteres en ambos documentos, para observar las diferencias entre algunos de ellos.

Ilustración 19: Carácter "g" en el documento de Cicerón

Ilustración 18: Carácter "g" en el documento de Persio

Ilustración 20: Carácter "e" en el documento de Cicerón

Ilustración 21: Carácter "e" en el documento de Persio

De entre las posibles mejoras que puede tener el entrenamiento al partir de un modelo pre-entrenado, lo que se pretende observar en estas pruebas es ver si el modelo entrenado a partir del modelo de Persio converge más rápido que el generado a partir de una red de neuronas cuyos pesos iniciales sean aleatorios. Las siguientes subsecciones describen como se prepararon los datos y los resultados obtenidos.

6.2. Preparación de los datos

Antes de pasar al proceso de entrenamiento se deben preparar los datos que necesitaremos. El conjunto de Persio ya está preparado de los puntos anteriores, por tanto, no será necesario realizar nada sobre estos datos. Los que sí se deberán preparar serán los datos del documento de Cicerón.

Estos datos fueron preparados de una manera similar a la de Persio, se contaba con la transcripción de todo el documento y el primer paso a realizar era incluir el ground truth de todos los datos.

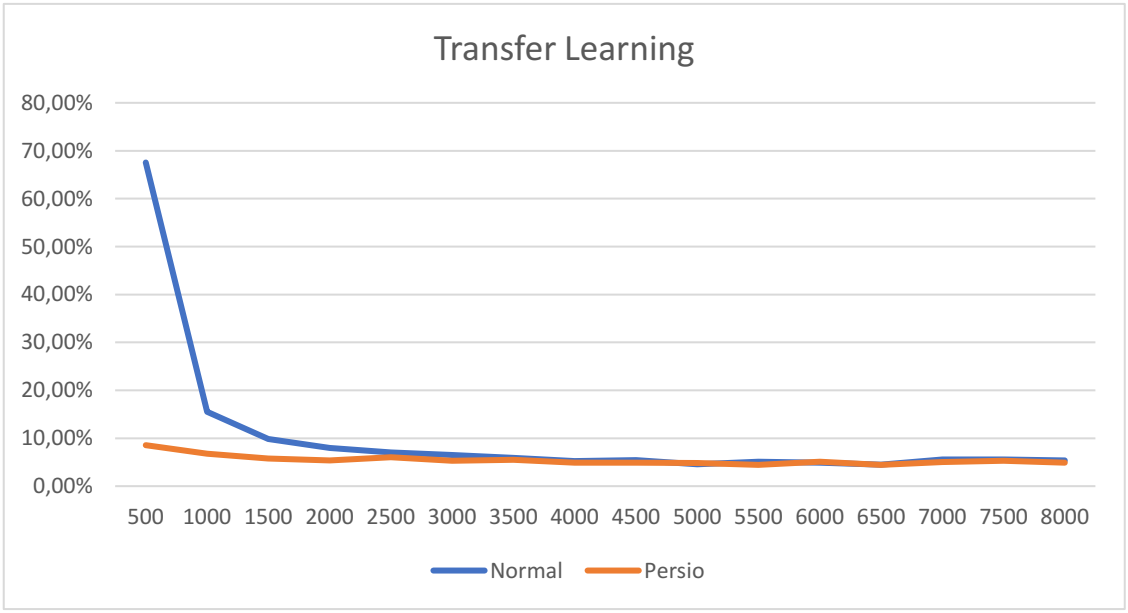
A continuación, había que decidir los conjuntos que emplearíamos del documento de Cicerón. Con el objetivo de simplificar van a utilizar los mismos parámetros (ciclos, neuronas ocultas y razón de aprendizaje) que para el documento de Persio, por tanto, se optó por no contar con un conjunto de validación. Es por eso que los de entrenamiento y de test serán divididos aproximadamente en un 80% de los datos para el de entrenamiento y el restante para el conjunto de test.

6.3. Resultados de Transfer Learning

Los parámetros elegidos para la red han sido los obtenidos en el capítulo anterior, es decir, en este caso una razón de aprendizaje de 0.001 y con 200 neuronas, además se guardarán modelos cada 500 ciclos.

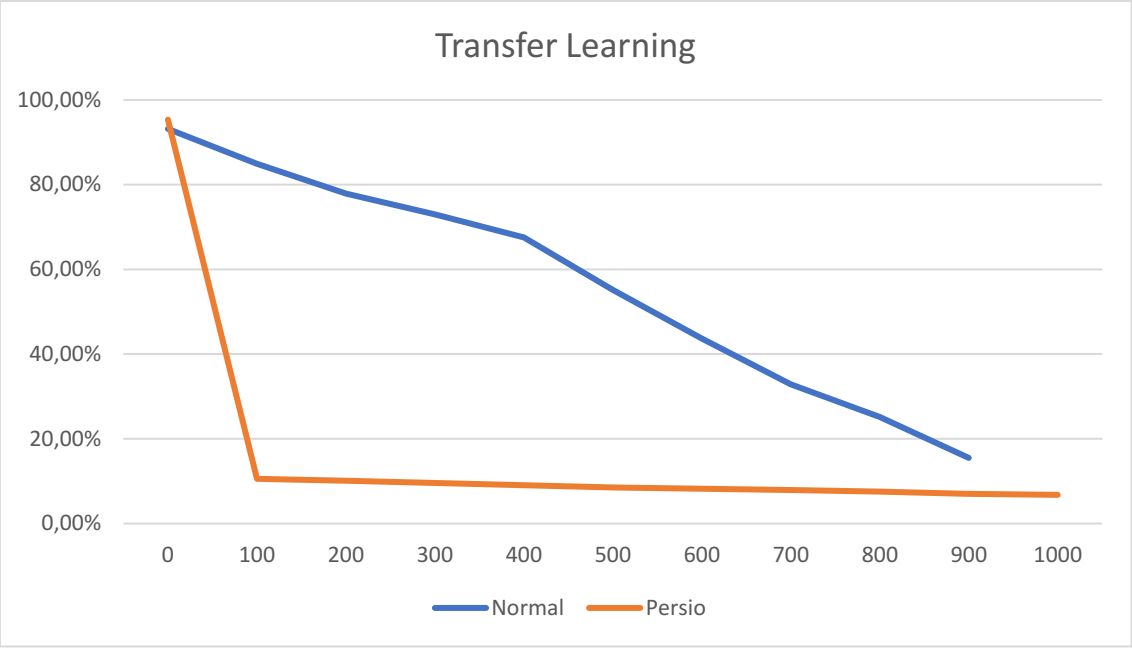
El procedimiento utilizado para el entrenamiento fue similar al realizado en los puntos anteriores, se realizó primero el entrenamiento de un modelo partiendo de pesos aleatorios (que es el procedimiento habitual) y después de uno partiendo del modelo de Persio (es decir, se parte de una red cuyos pesos ya no son aleatorios, sino pertenecientes al modelo pre-entrenado, y se continúa entrenando a partir de ese punto para el nuevo documento).

Como se puede ver en la gráfica 12 el modelo entrenado con la red de Persio empieza con un error mucho más bajo (en 500 ciclos tenemos la red normal con un error de 67,55% frente al 8,55% que obtenemos en el modelo pre-entrenado a partir del de Persio) y es a partir de los 2500 ciclos cuando los resultados de ambos modelos convergen y a partir de ahí hay una igualdad en los resultados.



Gráfica 12: Comparativa entre modelos transfer learning

Además, parece interesante observar en profundidad como estos modelos se comportan en los primeros 1000 ciclos del entrenamiento, que es donde se puede ver más diferencia entre los dos modelos en la gráfica anterior. Para ello en la gráfica 13 se realizó una comparación cada 100 ciclos en vez de cada 500.



Gráfica 13: Comparativa entre modelos transfer learning 1000 ciclos

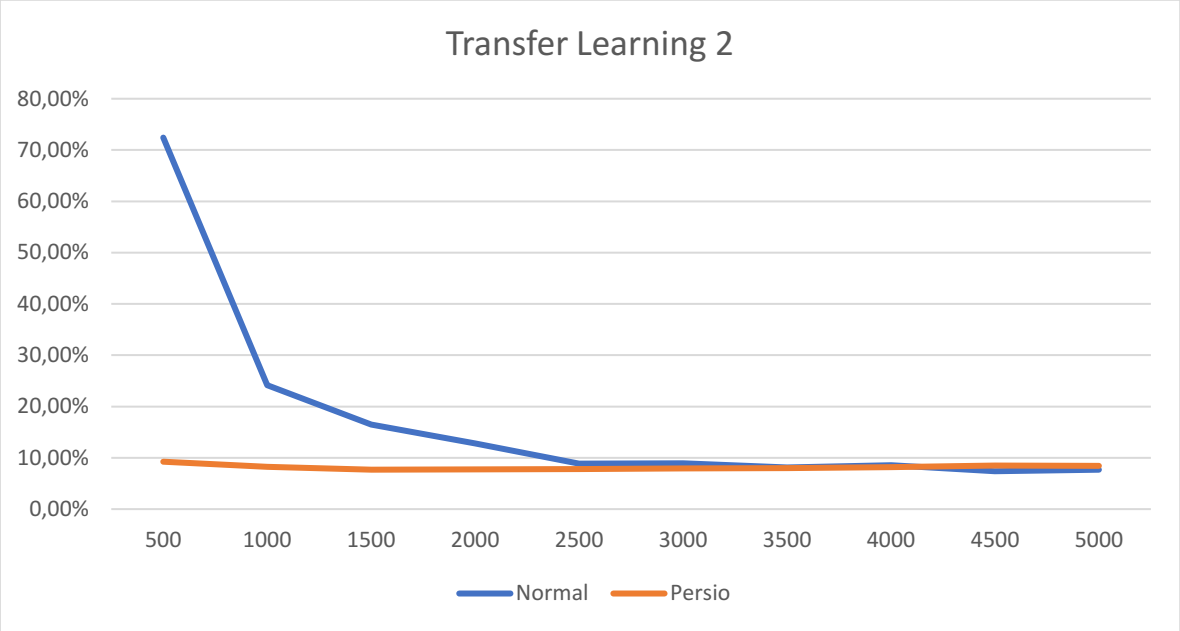
En la gráfica 13 se observa que el modelo entrenado a partir del documento de Persio después de entrenar sólo 100 ciclos ya obtiene unos resultados muy buenos, mientras que el modelo normal necesita bastantes más de los 1000 para obtener unos resultados equivalentes. También es interesante observar cómo sin entrenar la red de Persio con los datos nuevos, la red obtiene un error muy alto, pero como ya se ha observado, en sólo 100 ciclos adaptándose al documento nuevo, ya obtiene errores muy bajos.

Por último, después de ver los resultados obtenidos en la prueba anterior, parece interesante comprobar cómo se comportarían ambas redes con un número de datos de conjunto de entrenamiento de Cicerón mucho menor. En este caso, en vez de usarse un 80% del total será un 20%.

En la gráfica 14 puede verse los resultados obtenidos, en ella se pueden ver el porcentaje de error para cada uno de los conjuntos. Los resultados obtenidos ratifican lo visto en la prueba anterior. En ella podemos ver que el conjunto de Persio sigue obteniendo buenos resultados (aunque como es normal algo peores) y que hasta los 2500 ciclos no empiezan a estar igualados.

En estas pruebas también puede observarse que el entrenamiento del conjunto normal al contar con menos datos de entrada es notablemente peor en los primeros 2000 ciclos,

mientras que, aunque el modelo pre-entrenado ha empeorado también por la reducción del conjunto de entrenamiento, esta diferencia de resultados no es tan amplia.



Gráfica 14: Comparativa entre modelos transfer learning 2

Para cerrar este punto, se ha podido observar que, aun contando con grandes diferencias en los documentos utilizados, la red a partir del modelo de Persio, obtiene errores bajos con muy pocos ciclos.

7. PLANIFICACIÓN Y ENTORNO

En este capítulo se va a hablar sobre la planificación llevada a cabo durante este proyecto, el entorno socioeconómico del mismo y el entorno utilizado para la elaboración.

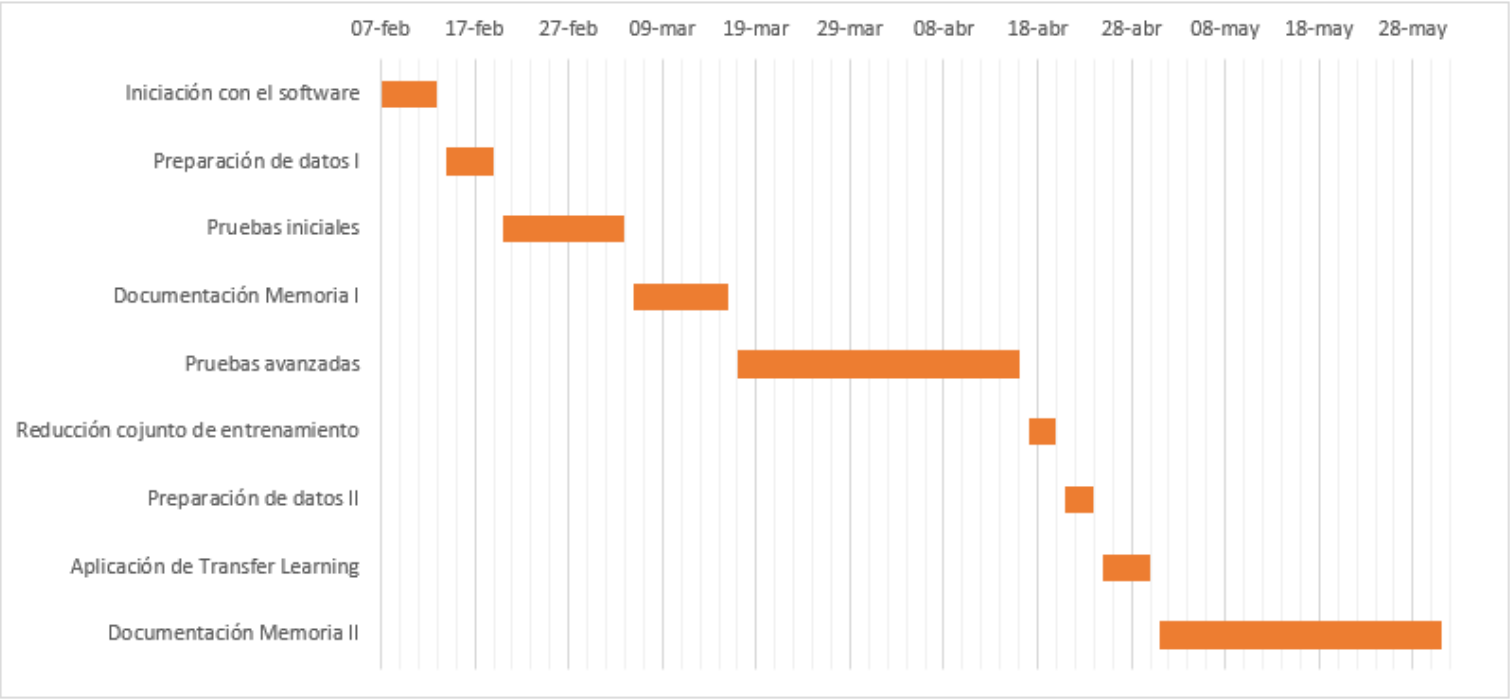
7.1. Planificación

El proyecto fue realizado siguiendo un desarrollo en cascada, por tanto, hasta terminar una fase no se debe seguir a la siguiente. Se establecieron diferentes divisiones en el proyecto que determinan distintas fases durante el mismo. En la tabla 5 se puede observar la planificación seguida durante el proyecto:

Tabla 5: PLANIFICACIÓN DEL PROYECTO

Nombre actividad	Fecha inicio	Duración en días	Fecha fin
Iniciación con el software	07-feb	6	13-feb
Preparación de datos I	14-feb	5	19-feb
Pruebas iniciales	20-feb	13	05-mar
Documentación Memoria I	06-mar	10	16-mar
Pruebas avanzadas	17-mar	30	16-abr
Reducción conjunto de entrenamiento	17-abr	3	20-abr
Preparación de datos II	21-abr	3	24-abr
Aplicación de Transfer Learning	25-abr	5	30-abr
Documentación Memoria II	01-may	30	31-may
Total Trabajo De Fin de Grado	07-feb	105	31-may

Se ha realizado el diagrama de Gantt sobre el proyecto para observar la planificación de este con las fases elegidas:



Gráfica 15: Diagrama de Gantt

Como se puede ver en la gráfica 15 estas fueron las diferentes fases de todo el proyecto, y hay que destacar algún detalle de esta.

Aunque el proyecto fue propuesto y aceptado en el mes de diciembre, no fue hasta inicios de febrero donde se inició el proceso de desarrollo de este. Hay que destacar que se han incluido dos fases de documentación de la memoria en unas fechas determinadas, aunque esta también fue realizada durante el resto de las fases.

En el proyecto, además se contó con un filólogo como experto en los distintos documentos antiguos, cuyo papel no fue solo ofrecer el material necesario, como son los documentos o las transcripciones de este, sino ayuda a la hora de iniciarse con el software de OCR utilizado.

La comunicación fue frecuente entre las personas implicadas en el proyecto a través de correos electrónicos donde se comunicaban los distintos avances y dudas al respecto. También se realizaron dos reuniones presenciales que tuvieron lugar en Madrid, las cuales fueron realizadas en diciembre a la hora de proponer este proyecto y fijar los primeros pasos a realizar, y una segunda, con el proyecto más avanzado, después de terminar las

pruebas iniciales y completar la primera parte de la memoria, en la que a partir de los datos dados se fijó el resto del proyecto y los pasos que queríamos dar.

7.2. Entorno

Todo el proyecto fue desarrollado en el ordenador personal del alumno, durante las distintas fases de este, el software se utilizó en diferentes medios dependiendo de la necesidad que se tratará de explicar a continuación.

El software fue durante todo el proyecto instalado en una máquina virtual local del ordenador, y es ahí donde se realizaron los primeros experimentos (capítulo 4), las pruebas (capítulo 5) y los experimentos de Transfer Learning (capítulo 6). El software funcionaba correctamente en este entorno y nos permitía ver la solución gráfica que ofrece Ocropus y que podemos ver en la ilustración.

Hay que decir también que se realizó una pequeña script en Python para contar los diferentes caracteres que contenía un fragmento de texto. Su principal objetivo era para determinar los conjuntos de entrenamiento, validación y test.

7.3. Entorno socioeconómico

7.3.1. Presupuesto

En este punto se va a documentar el coste del total del proyecto, separándolo por los diferentes tipos de costes.

En referente al presupuesto humano, hay que tener en cuenta que ha sido realizado por una persona, pero también han colaborado diferentes personas en el desarrollo del mismo quedando:

Tabla 6: COSTE HUMANO DEL PROYECTO

Persona	Coste por hora	Días	Coste total
Jose María Valls Ferrán	25 €	10	1.000 €
Ricardo Aler Mur	25 €	10	1.000 €
Manuel Ayuso Garcia	25 €	10	1.000 €
Luis Enrique Güendian García	25 €	105	10.500 €
		TOTAL	**13.500 €**

En la tabla 6 se puede observar el salario que corresponde a cada uno de los participantes en el proyecto, se ha asumido que cada persona ha trabajado 4 horas al día.

En cuanto al presupuesto material se puede observar en la tabla 7 los diferentes gastos que se han realizado durante el desarrollo del proyecto, además se ha contemplado el tiempo de vida del equipo y el tiempo usado durante el proyecto.

Tabla 7: COSTE MATERIAL DEL PROYECTO

Equipo	Coste	Uso	Depreciación	Coste imputable
Ordenador	1.000 €	4	24	166,67 €
			TOTAL	**167 €**

En lo referente al coste del software durante el proyecto se ha documentado en la tabla 8.

Tabla 8: COSTE DE SOFTWARE DEL PROYECTO

Producto	Coste
Microsoft Windows 10	50 €
Ocropus	0 €
Microsoft Word	0 €
TOTAL	**50 €**

Hay que destacar que la licencia de Ocropus es libre y por tanto no conlleva coste, y que la de Microsoft Word fue obtenida con la licencia de la Universidad.

Y por último otros gastos realizados en el proyecto:

Tabla 9: OTROS COSTES DEL PROYECTO

Descripción	Coste
Electricidad	40 €
Internet	100 €
TOTAL	**140 €**

Para terminar, en la tabla 10 se puede ver el coste total del proyecto, se ha tenido en cuenta tanto el riesgo de este como un 10% y el IVA.

Tabla 10: COSTE TOTAL DEL PROYECTO

Concepto	Coste
Salario	13.500 €
Equipo	167 €
Software	50 €
Otros	140 €
Coste Total	13.857 €
Riesgo	1.386 €
Coste Total con riesgo	15.242 €
IVA	2.744 €
Total	**17.986 €**

Por tanto, el coste total del proyecto será de 17.986€ contando tanto con el IVA como con un 10% de riesgo y considerando que al tratarse de un trabajo teórico no se pretende sacar beneficios de él.

7.3.2. Impacto

En este punto se va a tratar de hablar sobre el impacto que puede tener en la sociedad el trabajo realizado, así como de las posibles mejoras del mismo. El trabajo está realizado

principalmente para ayudar a los transcriptores con el manejo de la herramienta, y al tratarse de un trabajo teórico no será explotado por si mismo.

De este modo, lo que busca este trabajo es ayudar a los transcriptores a orientarse con el uso óptimo de la herramienta, contar con tamaños de muestra adecuados y como el uso del Transfer Learning puede acelerar los resultados.

Si hablamos de diferentes pruebas o trabajos que se pueden realizar encontramos las pruebas de "Data augmentation" que consiste en que con diferentes niveles de binarización de las páginas del documento se puede conseguir que con una misma línea se consigan diferentes imágenes. Es decir, con una única línea se podrían conseguir varias líneas aumentando el conjunto de datos disponible sin necesidad de transcribir más líneas. Además, al poder variar el umbral de binarización (parámetro incluido en el software de Ocropus) se puede simular diferentes presiones de la prensa de la imprenta.

De este modo, podemos obtener más ayuda e información sobre los datos necesarios para tener un conjunto de entrenamiento bueno con el menor esfuerzo posible.

8. **CONCLUSIONES**

En este proyecto se trataba de estudiar distintas cuestiones relacionadas con la digitalización de documentos antiguos, usando la herramienta Ocropus basada en redes de neuronas LSTM. Para la experimentación, se han usado documentos antiguos reales transcritos por un experto para este proyecto*: Las Satiras* de Persio (impreso en 1510) y los *Tópica* de Cicerón (impreso en 1506). Ambos fueron impresos en el mismo taller.

En concreto se han abordado tres cuestiones: el ajuste óptimo de la red para el aprendizaje, los tamaños de muestra mínimos para obtener buenos resultados, y el estudio de la técnica de Transfer Learning para ser capaces de entrenar una red a partir de una red pre-entrenada para otra tarea.

Con respecto a la primera cuestión, se ha comprobado que el ajuste de los parámetros principales de la red (número de ciclos, número de neuronas ocultas y razón de aprendizaje) es fundamental para obtener buenos resultados. Por ejemplo, el ajuste de parámetros permite pasar de un 5,83% de error (que es el peor resultado obtenido) a un 1,34%, lo que corresponde a una mejora relativa del 77%.

En segundo lugar, se ha comprobado que, con unas 160 líneas de entrenamiento, es suficiente para la red para obtener buenos resultados. Esto representa aproximadamente la mitad de las líneas de las que se partía. Este resultado puede ser útil para futuras transcripciones, puesto que, aunque el número de líneas necesario puede depender del documento concreto, proporciona una idea al experto que puede ahorrarse bastante esfuerzo de transcripción en el futuro, cuando necesite construir conjuntos de entrenamiento para las redes.

En tercer lugar, se ha aplicado la técnica de Transfer Learning pre-entrenando una red con el documento de Persio y haciendo un ajuste fino posterior para adaptarla al documento de Cicerón (que, aunque ha sido impreso en el mismo taller, tiene características ligeramente distintas). Los resultados muestran que la red pre-entrenada y posteriormente ajustada obtiene buenos resultados con muchos menos ciclos que una red que parta de cero. La red preentrenada también es menos sensible que la no preentrenada cuando se dispone de menos datos de entrenamiento. Por otro lado, también ha sido interesante comprobar como la red pre-entrenada sin ajuste posterior al nuevo documento, obtiene malos resultados. Es decir, la red pre-entrenada necesita el ajuste posterior.

Se puede concluir, por tanto, que los objetivos planteados fueron cumplidos, y que además se obtuvieron unos resultados prometedores para poder ser abordados en futuros trabajos.

9. BIBLIOGRAFÍA

[1] Serneguet, M. (2019). *19 BENEFICIOS DE LA DIGITALIZACIÓN DE DOCUMENTOS.* https://www.datadec.es/blog/factura-electronica-y-digitalizacion-certificada/19-beneficios-la-digitalizacion-de-documentos [Acceso 2 Jun. 2019].

[2] El libro antiguo, Manuel Pedraza García

[3] Ocadizdigital.es. (2019). [online] http://ocadizdigital.es/noticia/sociedad/hallan-el-incunable-m%C3%A1s-antiguo-en-castellano-de-los-siete-sabios [Acceso 15 Jun. 2019].

[4] Marsá Vilá, M. (2002). *La imprenta en La Rioja (siglos XVI-XVII)*. Madrid: Arco/Libros.

[5] Ejemplar BU Salamanca de la Edición de Arnau Guillén de Brocar

[6] https://www.ibiblio.org/pub/linux/docs/LuCaS/Presentaciones/200304curso-glisa/redes_neuronales/curso-glisa-redes_neuronales-html/x185.html. (2019). [online] [Accessed 15 Jun. 2019].

[7] Software-tecnico-libre.es. (2019). *Redes neuronales recurrentes y series de tiempo*. [online]http://software-tecnico-libre.es/es/articulo-por-tema/todas-las-secciones/todos-los-temas/todos-los-articulos/redes-neuronales-recurrentes-y-series-de-tiempo [Acceso 15 Jun. 2019].

[8] SearchDataCenter. (2019). *¿Qué es Aprendizaje profundo (deep learning)? –* [online] https://searchdatacenter.techtarget.com/es/definicion/Aprendizaje-profundo-deep-learning [Acceso 15 Jun. 2019].

[9] GitHub. (2019). *tmbdev/ocropy*. [online] https://github.com/tmbdev/ocropy [Acceso 5 Jun. 2019].

[10] Arxiv.org.(2019).[online] https://arxiv.org/ftp/arxiv/papers/1712/1712.05586.pdf [Acceso 15 Jun. 2019].

[11] Biografiasyvidas.com. (2019). *Biografia de Marco Tulio Cicerón*. [online] Available at: https://www.biografiasyvidas.com/biografia/c/ciceron.htm [Acceso 10 Jun. 2019].

[ ] Cis.lmu.de.(2019).http://www.cis.lmu.de/ocrworkshop/data/slides/m1-challenges.pdf [Acceso 10 Jun. 2019].

[ ] abc. (2019). *Hallan en Escocia el incunable más antiguo en castellano de «La historia de los siete sabios de Roma»*. https://www.abc.es/cultura/20140910/abci-incunable-siete-sabios-201409101255.html [Acceso 10 Jun. 2019].

ANEXO A. PORCENTAJE DE ERROR DEL MODELO EN EL CONJUNTO TEST

		500	1000	1500	2000	2500	3000	3500	4000	4500	5000	5500	6000	6500	7000	7500	8000	8500
50	0.0001*	82,76%	68,93%	50,87%	26,61%	17,58%	10,64%	9,26%	6,90%	6,29%	6,20%	5,45%	4,45%	4,18%	4,19%	3,67%	3,61%	4,20%
	0.0005	59,90%	23,15%	8,97%	7,58%	7,93%	10,17%	9,94%	24,50%	9,92%	8,30%	5,40%	4,89%	4,47%	9,42%	4,91%	4,23%	4,52%
	0.001	36,63%	11,04%	8,26%	11,10%	7,06%	6,74%	5,83%	7,50%	20,49%	15,58%	18,17%	10,01%	41,62%	--	--	--	--
100	0.0001*	77,36%	55,19%	33,38%	10,71%	5,26%	4,60%	4,20%	3,97%	3,47%	3,09%	2,68%	2,58%	2,62%	2,45%	2,73%	2,17%	2,24%
	0.0005	24,43%	6,55%	5,38%	3,70%	2,60%	2,79%	2,18%	2,13%	1,93%	1,98%	1,76%	2,16%	1,70%	1,91%	2,40%	1,74%	1,87%
	0.001	22,22%	5,71%	6,35%	4,01%	4,14%	3,04%	4,65%	4,95%	6,45%	8,68%	6,40%	26,82%	13,23%	13,00%	17,57%	6,26%	8,22%
200	0.0001*	63,27%	16,42%	6,58%	4,72%	3,51%	2,73%	2,62%	2,01%	2,05%	1,86%	1,78%	1,83%	1,84%	1,74%	2,14%	1,61%	2,01%
	0.0005	23,22%	6,58%	3,76%	2,47%	1,80%	1,75%	1,84%	2,21%	1,88%	1,41%	1,51%	1,61%	1,64%	1,34%	1,33%	2,13%	5,21%
	0.001	12,07%	4,65%	3,61%	2,35%	14,81%	27,90%	15,86%	42,33%	60,56%	--	--	--	--	--	--	--	--

ANEXO B. PORCENTAJE DE ERROR DEL MODELO SEGÚN EL NÚMERO DE DATOS EN EL CONJUNTO DE ENTRENAMIENTO

	300	280	260	240	220	200	180	160	140	120	100	80	60	40
Validacion	4,65%	4,85%	5,03%	5,60%	4,99%	5,37%	5,73%	5,57%	6,18%	6,57%	8,76%	9,36%	10,83%	24,15%
Test	1,34%	1,75%	1,83%	2,25%	1,84%	1,76%	2,05%	1,94%	2,33%	2,63%	2,96%	3,48%	4,65%	17,03%

ANEXO C. COMPARACIÓN DE DOS MODELOS SEGÚN SI SE APLICA TRANSFER LEARNING O NO

	500	1000	1500	2000	2500	3000	3500	4000	4500	5000	5500	6000	6500	7000	7500	8000
Normal	67,55%	15,50%	9,86%	7,98%	7,01%	6,50%	5,92%	5,25%	5,44%	4,55%	5,08%	4,93%	4,49%	5,58%	5,58%	5,35%
Persio	8,55%	6,77%	5,78%	5,39%	6,03%	5,30%	5,50%	4,88%	4,90%	4,84%	4,44%	5,08%	4,44%	5,04%	5,33%	4,92%

ANEXO D. RESUMEN EN INGLÉS

1. INTRODUCTION

This paper deals with the problem of the digitization of old documents, which has specific characteristics quite different from the digitalization of modern documents.

There are digitization tools oriented to this type of documents based on Neural Networks that use deep learning techniques for their learning. One of the most used tools is Ocropus, which in turn uses LSTM type networks. In this work, different aspects of the digitalization of old documents will be studied with the Ocropus tool.

A first question is related to the fact that LSTM networks have several adjustable parameters that influence their learning and as a second question the technique known as Transfer Learning will be applied, which consists of using pre-trained networks in a task (a document) as a point of game to solve another (another slightly different document, in this case).

1.1. Objectives

The main objective pursued with this project is the study of the behavior of networks of neurons from different parameters for the digitalization of these documents. In addition to the main objective there are several secondary ones:

- Use of a tool specialized in scanning old documents.

- Optimization of this tool by adjusting its parameters

- Determination of the size of the minimum training set capable of producing acceptable results

- Study of Transfer Learning techniques to try to improve the learning of the network, both in computational cost and in results.

With these objectives it is sought to bring the optical character recognition a little closer

OCR, to this type of documents to the maximum possible effectiveness, which clearly is not what is currently obtained.

2. STATE OF ART

In this chapter of the report we will talk about the current status of the main theme of the project, historical documents and neural networks and also the digitalization of current documents and historical documents.

2.1. Documents of the project

At this point we are going to talk about the type of documents that will be used in this project. These documents are called old documents and represent those documents printed before the invention of the automatic press in the nineteenth century. [3]

The printing was a great revolution, was invented by Gutenberg, when around 1450 printed his first book, the Gutenberg Bible. The printing arrived in Spain in 1472.

The books printed during the fifteenth century are known as incunables of which currently some 300,000 copies are preserved. These documents have some differences from the current ones that cause problems when digitizing them, which are:

- Different versions: One of the problems is that for the same work arise different versions that may end up differing enough, due to errors of the printer or printing defects.

- Glosses: These are manuscript comments in the margins of a book, which can be found for different reasons, from explaining the meaning of the text in its original language to comments on it.

- Character defects: Being a printing company, there is great diversity in the characters that make up these forms. It can be assumed that the same character is exactly the same throughout the text, but this is not always the case, since the printing press often ran out of ink or had excess, making small differences between the same characters within the document.

2.2. Artificial neural network

Artificial neural network are models formed by elements, called neurons that behave by trying to imitate the basic functions of human neurons. Its main characteristic is that given an input set the networks can be adjusted to produce the desired output for a given problem.

An important phase of the artificial neural network is their learning, in which they try to train the neural network to generate the best possible output for each data.

This learning can be of two types:

- Supervised Learning: This type of learning is characterized because the training is controlled by an external agent, which determines the output that the network should generate for the determined inputs.

- Non-Supervised Learning: This type of learning is characterized because the output represents the similarity or similarity between the current entry and the previous ones. The data does not contain information regarding the desired output.

The networks that are used in the software that we will use in the project are the multilayer perceptron and the recurrent LSTM networks, therefore, they will be explained below.

2.2.1. Multilayer Perceptron

It can be said that the multilayer perceptron is one of the most used architectures for solving problems, among other things due to its easy use. This architecture allows solving non-linear regression problems. Its architecture is made up of different layers:

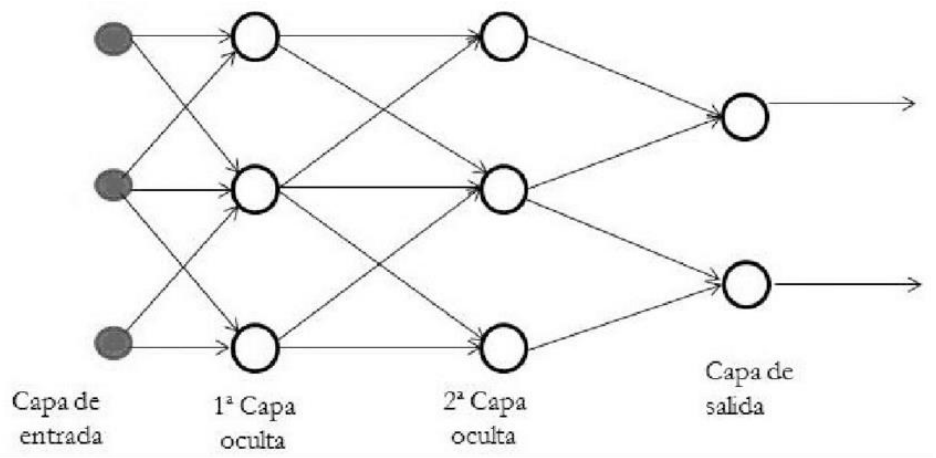


Illustration 22: Multi-layer perceptron structure

This architecture is a universal approximation, that is, it can approximate any continuous function thanks to its innovative training algorithm (retropropagation algorithm).

2.2.2 Recurrent neural network

Recurrent networks are neural networks that are characterized by the use of the network output as one more entry, allowing the network to have memory. In other words, its main characteristic is that the information can persist by introducing loops in the network diagram, that is, they can remember what the previous states were and use this to determine the next state.

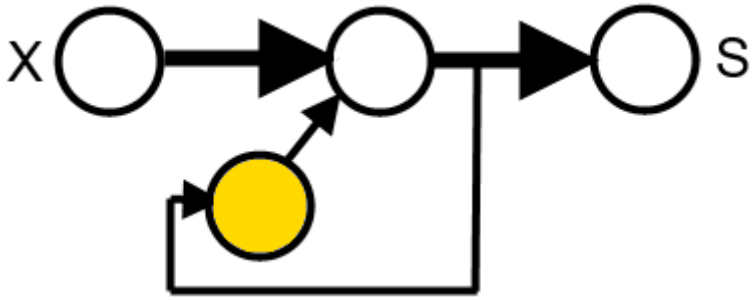


Illustration 23: Representation of recurrent neural network

When introducing these connections, the number of weights increases complicating the learning with respect to other types of networks. In addition, the architecture of these networks of neurons has many hidden layers.

The problem is that the backpropagation algorithm works well with one layer, but with many no, this is called the gradient problem and to solve this recurring LSTM networks arose. Recurrent LSTM networks are characterized by the fact that they can learn long dependencies for what they remember in the longer term.

3. EXPERIMENTS PERFORMED

In this chapter we will proceed to study the training of the network to solve the digitization problem addressed in this work.

3.1 Data preparation

In these tests it will be used as the base document *The Satires* of Persio. This document consists of 22 pages and about 700 verses.

To have the data in the form we need we must separate the image of each page in an image for each line, but with old documents we find several problems such as noise, glosses and capital letters.

Then, these data must be divided between the sets used, in this case we will use the training, test and validation sets. So that the test set is completely independent of the learning process, the training set will be used to build different models (with different parameters) and the validation set will be used to compare them and choose the best one.

3.2. Tests

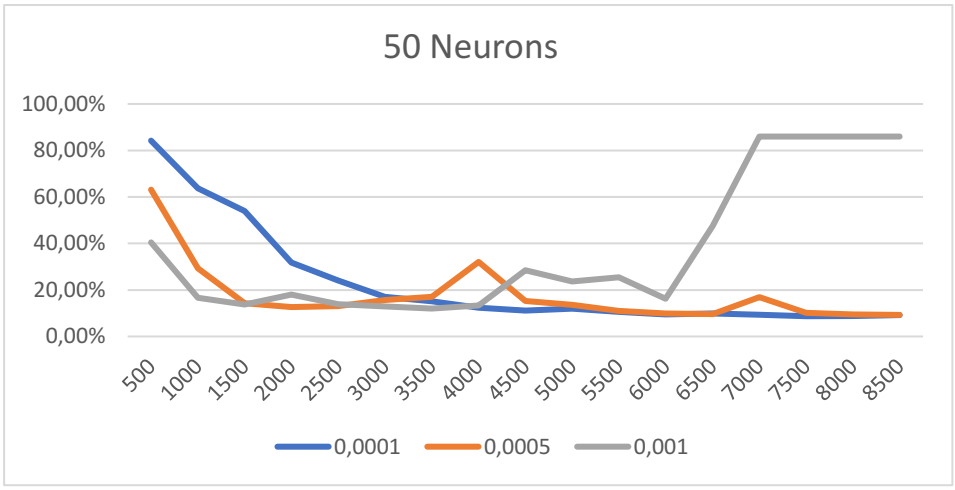
For the tests carried out, we will use different values for the parameters available in the software used. These parameters are:

- Number of cycles: The number of cycles used in these tests will be 8500 cycles and a model will be saved every 500 cycles.

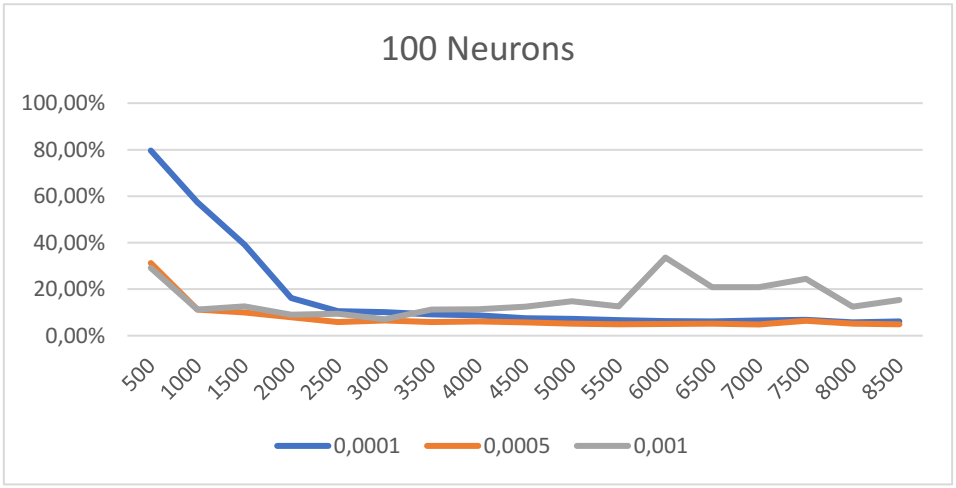
- Hidden neurons: The number of neurons controls the complexity of the model. In this case we will use as number of hidden neurons 100, 200 and 300 neurons.

- Learn rate: Default 0.0001. They are used as values 0.0001, 0.001 and 0.005. Initially 0.01 was used but it was too high a value that gave poor results.

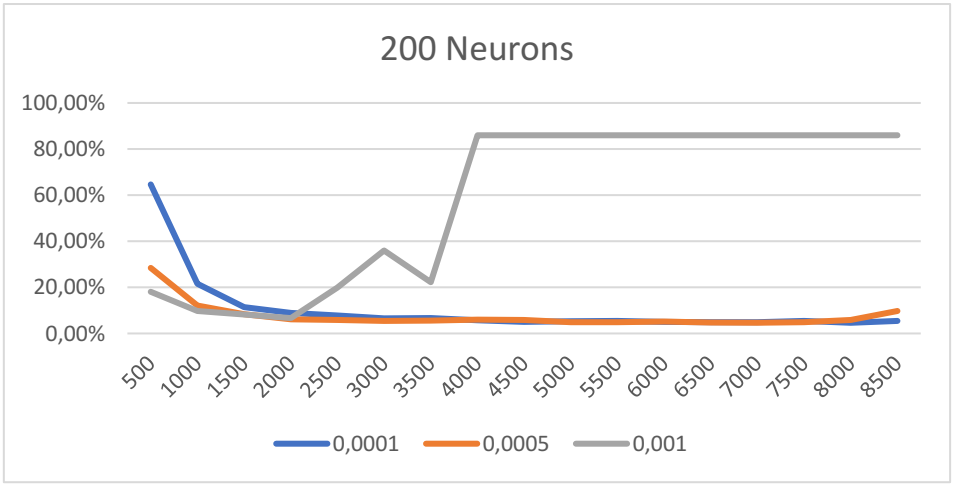
In graphs 16,17 and 18, we can see the evolution of the error in the validation set with 50,100 and 200 neurons, respectively. In each graph we find 3 lines that represent the three values of the learning ratio used.



Graph 16: Percentage of error in the validation set for 50 neurons



Graph 17: Percentage of error in the validation set for 100 neurons



Graph 18: Percentage of error in the validation set for 200 neurons

In these graphs we can see some general details:

- It seems that 8000 cycles are excessive for values with a learning ratio of 0.001, since it reaches its optimal point much earlier

- As you can see in the three graphs in many of the models you can reach an error rate of around 5%

- You can see how for a lower learning ratio, later on we reach the optimal values of the model, but we arrive more gradually, at smaller errors, and in general, it does not happen that the error starts to get worse

3.3. Choose the optimal parameters

In this section we will try to choose the best model based on the specific validation errors. A summary table has been prepared with the best models for each set of values.
In the table 11 you can see all validation errors for all parameter combinations (hidden neurons, learning ratio and number of cycles). In it, the cycles column indicates when it reaches the minimum error in validation.

Table 11: PERCENTAGE OF ERROR BY MODEL

Neurons	L_Rate	Cycles	%ValidError	%TestError
	0.0001*	7500	8,70%	3,67%
50	0.0005	8500	9,27%	4,52%
	0.001	3500	11,99%	5,83%
	0.0001*	8000	5,73%	2,17%
100	0.0005	7000	4,82%	1,91%
	0.001	3000	7,05%	3,04%
	0.0001*	8000	4,64%	1,61%
200	0.0005	7000	4,65%	1,34%
	0.001	2000	6,68%	2,35%

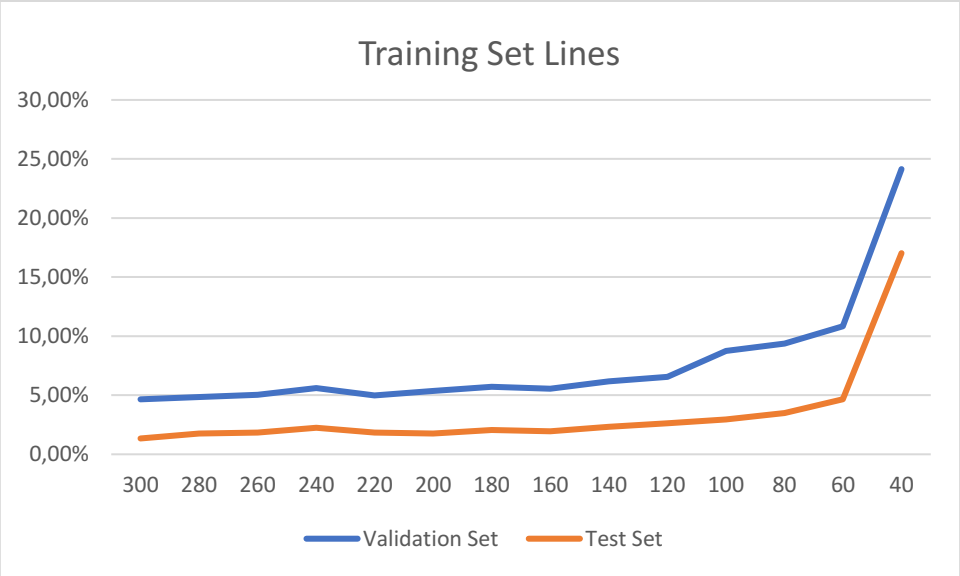
We can see that there is a relationship between the results obtained for the same set of values for the parameters is fulfilled. This fact can be seen in that in both errors the relationship of the three best sets of parameters is maintained.

Due to the equality of the three best models and not noticing any notable difference between them, those whose validation error is lower will be chosen as optimal parameters. In this case, there will be 200 neurons and 0.0001 as a learning reason for future sections.

3.4. Determine the relationship between size of the training and error assembly

Once the optimal parameters have been chosen, the next objective is to carry out a study about the minimum amount of training data to obtain good results, since the cost of transcribing the data is high both at the computational level and at the human level.

Returning to the sets used in the previous tests, the training set consisted of 60% of the total data, and the objective is to verify what happens with the error if a smaller size is used. This 60% consists approximately of a total of 300 data or lines of the document, which will be reduced from 20 in 20.



Graph 19: Error percentage of the model according to the number of data in the training set:

In summary, as you can see in the 19 graph it seems that from 160 lines the error is out of control, therefore, it can be considered that to obtain useful results it is necessary to train about 160 lines of each document.see how this affects the model.

4. TRANSFER LEARNING

Transfer learning is a type of learning characterized by the use of a previously trained model with data from another document. At this point we will try to relate this concept with the used software used.

In this section, a similar study of digitization of a new document is going to be carried out. The objective is to compare the results obtained by doing the same type of training as in previous points with another, using a model of the previous section previously trained to start from it.

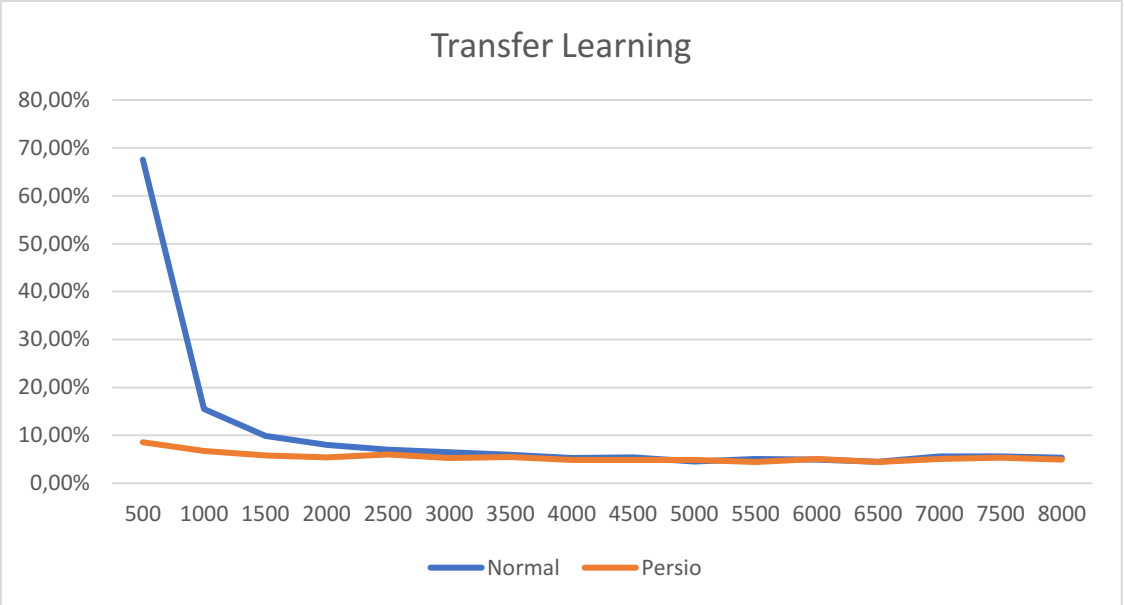
Among the possible improvements that training can have when starting from a pre-trained model, what we intend to observe in these tests is to see if the model trained from the other model converges faster than the one generated from a network of neurons whose initial weights are random(standard procedure) .

The data should be prepared in the same way as in Section 3 with the Persio document (line separation, enter the transcription for the lines). Once done we will train the network and the results obtained can be seen in the next point.

4.1. Results of Transfer Learning

The parameters chosen for the network have been those obtained in the previous chapter, that is, in this case a learning ratio of 0.001 and with 200 neurons, in addition models will be saved every 500 cycles.

As can be seen in the graph 20, the trained model with the Persio network begins with a much lower error (in 500 cycles we have the normal network with an error of 67.55% compared to the 8.55% we obtain in the model pre-trained from Persio's) and it is from the 2500 cycles when the results of both models converge and from there there is an equality in the results.



Graph 20: Comparison between transfer learning models

As can be seen in the graph 20, the trained model with the Persio network begins with a much lower error (in 500 cycles we have the normal network with an error of 67.55% compared to the 8.55% we obtain in the model pre-trained from Persio's) and it is from the 2500 cycles when the results of both models converge and from there there is an equality in the results.

It is also interesting to observe how without training the Persio network with the new data, the network gets a very high error, but as it has already been observed, in a few cycles it adapts to the new document and already gets very low errors.

To close this point, it has been observed that, even with large differences in the documents used, the network from the Persio model, gets low errors with very few cycles.

5. CONCLUSIONS

In this project, the aim was to study different issues related to the digitization of old documents, using the Ocropus tool based on networks of LSTM neurons.

In particular, three issues have been addressed: the optimal adjustment of the network for learning, the minimum sample sizes to obtain good results, and the study of the Transfer Learning technique.

Regarding the first question, it has been verified that the adjustment of the main parameters of the network (number of cycles, number of hidden neurons and learning ratio) is fundamental to obtain good results.

Secondly, it has been proven that with about 160 training lines, it is sufficient for the network to obtain good results. This represents approximately half of the lines from which it was split.

Third, the Transfer Learning technique has been applied pre-training a network with the Persio document and making a fine adjustment later to adapt it to the Cicero document (which although it has been printed in the same workshop, has slightly different characteristics). The results show that the pre-trained and subsequently adjusted network obtains good results with many fewer cycles than a network that starts from scratch.

It can be concluded, that the proposed objectives were met, and that some promising results were obtained to be addressed in future work.